IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 18, 2019, accepted November 17, 2019, date of publication November 26, 2019,
date of current version December 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2955983

An Imbalanced Big Data Mining Framework for Improving Optimization Algorithms Performance

ESLAM MOHSEN HASSIB[1], ALI IBRAHIM EL-DESOUKY[1], EL-SAYED M. EL-KENAWY[2], AND SALLY M. EL-GHAMRAWY[3]

[1]Computer Engineering and Systems Department, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt
[2]Department of Computer and Systems Engineering, Delta Higher Institute for Engineering and Technology (DHIET), Mansoura 35111, Egypt
[3]Head of Communications and Computer Engineering Department, MISR Higher Institute for Engineering and Technology, Mansoura 35111, Egypt

Corresponding author: Eslam Mohsen Hassib (eslamhassib21@hotmail.com)

ABSTRACT Big data is an important factor almost in all nowadays technologies, such as, social media, smart cities, and internet of things. Most of standard classifiers tends to be trapped in local optima problem when dealing with such massive datasets. Hence, investigating new techniques for dealing with such massive data sets is required. This paper presents a novel imbalanced big data mining framework for improving optimization algorithms performance by eliminating the local optima problem consists of three main stages. Firstly, the preprocessing stage, which uses the LSH-SMOTE algorithm for solving the class imbalance problem, then it uses the LSH algorithm for hashing the data set instances into buckets. Secondly, the bucket search stage, which uses the GWO for training bidirectional recurrent neural network BRNN and searching for the global optimum in each bucket. Lastly, the final tournament winner stage, which uses the GWO+BRNN for finding the global optimum of the whole data set among all global optimums from all buckets. Our proposed framework LSHGWOBRNN has been tested over 9 data sets one of them is big data set in terms of AUC, MSE, against seven well-known machine-learning algorithms (Naive Bayes, Random Tree, Decision Table, and AdaBoostM1, WOA+MLP, GWO+MLP, and WOA+BRNN), then, we tested our algorithm over four well-known data sets against GWO+MLP, ACO+MLP, GA+MLP, PSO+MLP, PBIL+MLP, and ES+MLP in terms of classification accuracy and MSE. Our experimental results have proved that our proposed framework LSHGWOBRNN has provided high local optima avoidance, and higher accuracy, less complexity and overhead.

INDEX TERMS Grey wolf optimizer, neural network, big data mining, deep learning, imbalanced data sets, optimization.

I. INTRODUCTION

The rapid growth of smart devices, internet of things, smart cities and massive number of sensors networks are leading the world to be flooded by a gigantic amount of data generated from numerous sources, such as social networks, sensor networks data, video broadcasting sites, bioinformatics, internet marketing and more. Extracting knowledge from such vast data sets is considered as one of the biggest challenges for most of traditional machine learning techniques [1]. As a result of the difficulties and challenges of processing, analyzing and extracting useful information from such massive amounts of data, a new concept has arrived ''big data'' [2]. The major issue with big data can be defined by using the big data main four features, massive volume, enormous variety, vast velocity, and veracity, which can be called the 4V model [3]. The most important feature among the 4V features of big data and its biggest issue is the massive-volume that stands for the gigantic amount of data collected [4].

The main sources for big data generated nowadays are numerous and diverse, but three of them are considered as the main source of big data, Internet of things, Online social networks, and biomedical data [5]. Facebook with 1.33 billion active users, then, Google+ with more than 255 million users, then, Twitter also with more than 220 million users and LinkedIn with more than 180 million users [6], are creating more than 20 Terabytes (TB) every single day [7]. Also, smart cities with millions of sensors are creating a huge amount of data that needs to be processed and stored [8]. In the e-health bioinformatics field, millions of patient's records and vast

The associate editor coordinating the review of this manuscript and approving it for publication was Yanbo Chen.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

amount of data generated by sensors and actuators worn by patients are created, stored, and processed per day [9]. E-mail service providers are exchanging more than 210 billion messages every day [10]. Moreover, the Internet-based devices number is expected to grow up to 100 billion devices by 2020 [11].

Another issue when dealing with big data is its tremendous variety, since big data is involved in many different data sources [12], hence, it includes most of data formats available nowadays such as; texts, images, videos, and so on. The third issue of big data is its vast velocity, which refers to the big data fast generation rate. Accordingly, effective and efficient real time processing techniques are essential [13]. The big data real time processing is crucial for many web applications nowadays, such as cyber security, online money transfer transactions, and electronic commerce. Unfortunately, not all big data available nowadays can be processed efficiently for extracting meaningful information due to the lack of resources and poor analysis tools [14]. Hence, a big percentage of big data that is to be processed is either delayed, neglected or deleted. As a result, a huge percentage of networking power consumption, storage and bandwidth are wasted. Finally, the fourth and last issue with big data is its massive veracity which refers to the existence of vast number of inaccurate objects, incomplete objects, noisy objects, and redundant ones [15].

Furthermore, most big data nowadays suffers from a critical problem known as class imbalance problem. A data set is considered imbalanced, if the number instances in one class, is massively outnumber the instances in the other class. The main issue with the class imbalance problem is that the results will be biased toward the majority class which could provide us with inaccurate classification results and lead us to take wrong decisions [16]. This issue happens because most of classifiers does not usually consider the data distribution when reducing the global parameters, for instance the error rate [17]. So, a preprocessing phase is needed for solving the problem of imbalanced classes in such datasets before continuing toward the classification phase [18].

One of the most promising solutions for big data issues is Deep learning due to its huge capability of extracting the meaningful information from massive data sets [19]. Before deep learning, only a few classification techniques were able to handle big data, but today with the great revolution of computer hardware, particularly, in graphics cards a massive computational power provided by GPUs has become available even for ordinary people not only companies. Online social networks can be considered as one of best examples for adopting deep learning for handling big data sets, video games development, and artificial intelligence (AI) [20].

Nowadays, Deep learning have been applied to many fields such as machine learning, computer vision, social network filtering, speech recognition, machine translation, medical image analysis, bioinformatics, drug design and game industries where they have produced excellent results compared to traditional machine learning techniques [21]. Recently, in September 2018, Nvidia the famous graphics cards manufacturer produced a new graphics cards series, the RTX series. RTX cards uses a deep learning technique called DLSS (Deep Learning Super Sampling) for increasing the games fps (Frames per Second) and improving the games performance, which hugely increased the games performance, sometimes more than 90%[22].

Artificial Neural Networks (ANNs) are emulating the way of information processing and communication distributed nodes in biological nervous systems. ANNs mainly consists of a group of connected units called artificial neurons, the connections between neurons are called (synapse), and it can transmit signals between neurons [23]. The receiver neuron is called (postsynaptic), and it can process the signals, and then send signals downstream to other neurons connected to it. Neurons usually have a state, and it is commonly represented by real numbers between 0 and 1. The strength of the signal that neurons send downstream can be increased or decreased, and it is mainly depending on the weights of the neurons and synapses, which varies as learning proceeds. Neurons are usually organized in layers; the type of input processing operations performed in each layer can vary between different layers [24]. Signals are traveling from the first layer (input) to the last layer (output), sometimes, its travers the layers multiple times.

In the last years, ANNs have been applied to many fields including regression, speech recognition, and machine learning algorithms [25]. The ANNs performance is hugely affected by the learning process and parameters optimization. Multilayer perceptron (MLP) is one of the most commonly applied ANNs. Supervised learning techniques can be divided into two main categories: stochastic techniques and gradient-based [26]. A typical example of gradient-based techniques is the back-propagation algorithm and its variants [27]. Nevertheless, gradient-based techniques mainly suffer from three main issues: its convergence is very slow, it is highly dependent on initial values [28], and it can be trapped easily in local optima [29]. Local optima can be defined as the best solution within a group of neighboring solutions that may be incorrectly considered as the global optimum, which can be defined as the optimal solution among all possible solutions in the whole dataset [30].

In this paper a novel imbalanced big data mining framework for eliminating the local optima problem consists of three main stages is presented. The first stage is the preprocessing stage, which uses the LSH-SMOTE algorithm for solving the class imbalance problem, then it uses the LSH (Locality Sensitive Hashing) algorithm for hashing the data set instances into buckets, for creating subsets data for simplifying the search of global optimum in each bucket. The second stage is the bucket search stage, which uses the GWO (grey wolf optimizer) for training and adjusting the weights and biases of Bidirectional Recurrent Neural Network (BRNN) and searching for the global optimum in each bucket, composing a new data set from the best 5 search agents in each bucket called Final dataset. The third

stage is the final tournament winner stage, which uses the GWO+BRNN for finding the global optimum of the whole data set among all global optimums from all buckets in the Final dataset.

The main reason for using the Grey wolf Optimizer (GWO) in this paper, is its high local optima avoidance ability that will help us optimizing the BRNN parameters, and solve the BRNN unstable gradient problem, which accordingly will improve the classification results. GWO is one of the recent meta-heuristic optimization algorithms, and it was proposed by Seyed Ali Mirjalili in 2013 [31]. The GWO algorithm simulates the hierarchy of leadership and hunting behavior of grey wolves in nature. In [31], The GWO was tested and compared against different well-known optimization algorithms such as GSA, PSO, EP, ES and DE algorithm to solve 29 well-known mathematical optimization problems, and three classical engineering design problems such as (the design of tension/compression spring, the design of welded beam, and the design of pressure vessel). As reported in [31], the GWO algorithm provided very competitive and excellent results compared to these well-known meta-heuristics algorithms.

The rest of this paper is organized as follows: Section 2 illustrates the background. Section 3 provides and explains in details the component of the proposed framework LSHGWOBRNN. Section 4 demonstrates and discusses the experimental part. Finally, the conclusion is presented in Section 5.

II. BACKGROUND

In this section, we will explain in details; imbalanced data sets, Locality Sensitive Hashing (LSH) algorithm, Recurrent neural networks, and finally Grey wolf optimizer.

A. IMBALANCED DATASETS

The dataset is considered to be imbalanced if the number of samples in the majority class is overwhelming the samples of minority class [32]. For instance, the population of patients constitutes only a small part compared to the normal healthy persons. The more dangerous diseases have even rarer numbers of cases, such as AIDS and cancer. Frankly speaking, it is very dangerous to identify one of patients with infectious diseases as a healthy person and vice versa [33]. That creates the imbalanced data set when we try to classify such data which causes over-fitting the majority classes and could lead to biases classification results and wrong decisions. Traditional classifiers are reporting very bad results when they are applied to imbalanced datasets [34]. For instance, the classifier could report a very good performance on the majority class but, on the other hand, it could report a very bad performance on the minority class, since they consider a balanced data distribution.

For solving the imbalanced classes problem, many techniques have been proposed for solving this issue. Sampling is one of the proposed techniques for preprocessing the imbalanced classes problem [35]. Sampling is mainly based on a simple idea which is, achieving the balance between data set classes. Oversampling and under-sampling are the main methods of sampling [36]. Under-sampling is achieving the data distribution balance by removing some instances from the majority class. In contrast, oversampling is trying to achieve the balance by duplicating the minority class instances. Unfortunately, both algorithms are suffering from severe disadvantages. Under-sampling may neglect some important instances, which accordingly, may affect the classification algorithms performance. Conversely, oversampling may create unnecessary minority class instances, which may consequently increase the algorithm running time. Generally, Duplicating the classes instances can lead to over fitting. for solving the over fitting problem, Chawla, Bowyer, and Hall produced the SMOTE algorithm [37]. The basic idea of SMOTE algorithm is generating synthetic instances of the minority class by using the attribute domain instead of the instance domain by creating synthetic instances of the minority class.

B. LOCALITY SENSITIVE HASHING (LSH)

Locality sensitive hashing (LSH) technique is one of the best techniques for finding related instances in a very short time compared to other techniques in massive data sets [38]. This algorithm belongs to randomized class algorithms. Randomized algorithms do not necessarily guarantee an accurate answer but instead, give us a promise of returning the right answer or one very adjacent to it. By performing additional computational iterations, we can achieve higher probability. There are many issues in the real-world applications today that require finding similar samples. Usually, we can solve such a problem by finding the nearest neighbor to an item in some dimensional space [39]. This solution looks easy, but when the dataset is massive and the instances are too interfered and complicated, the time required for processing is increasing linearly with the number of instances and the complexity of the classes inside the dataset.

Theoretically, we can solve this problem by reiterating through every object in the dataset and computing the distance to the query object. Nevertheless, our datasets may consist of billions of instances; a vector that contains hundreds of dimensions describes each object. Therefore, it is very important to find a method that does not use linear search method in the dataset. Modern methods to achieve this solution include trees and hashes [40]. A Tree is starting from the top node of the query then descending down constructing a tree of leaf's, unfortunately, the time required for constructing such a tree is O (log N), where N is the number of instances in the dataset. In one-dimensional space, it will be considered as a linear search, but in multi-dimensional space, this requires constructing the k-d tree. The main drawback for k-d trees is that its performance goes down when the number of dimensions increases hugely; accordingly, we cannot use it for massive datasets with complex dimensions [41].

Hashing basic idea is to build a hash table for mapping between keys (indexes) and an array of buckets that contains the values of the dataset. A well-designed hash table will

allow us to find a value in O (1) with O(K) memory, where K is the number of keys in the table [42]. The main advantage for hashing over other techniques like Trees is the speed, especially when the number of dimensions and entries is too high. For such massive and high dimensional datasets, locality sensitive hashing (LSH) is one of the best techniques to use for retrieving samples that are similar to a query item. In these searches, it can significantly diminish the computational time, at a cost of losing small percentage of precision and accuracy [43]. Indyk and Motwani introduced the idea of locality sensitive hashing (LSH) in [44]. LSH function families have a very important property, that the probability of collision between near instances are much higher than far ones [45].

At the beginning, the dot product (DP) (scalar projection) is applied to ensure that all near instances will fall into the same bucket according to the following equation:

$$DP\left(\vec{q}\right) = \vec{q} \cdot \vec{v} \tag{1}$$

where $\vec{q}$ is a query point, and $\vec{v}$ is a Gaussian random vector, this scalar projection is then quantized into a set of buckets, with a width w, with one assumption that adjacent items in the original space will fall into the same bucket using the following equation:

$$h^{v,z}\left(\vec{q}\right) = \frac{\vec{q} \cdot \vec{v} + z}{w} \tag{2}$$

where w is the width of each quantization bucket and z is a Random variable distributed from zero to w uniformly for reducing the quantization error and maintain the same performance [46]. At the end of projection and quantization operations, each instance in our dataset should be placed in a bucket described by d integer indices [47]. An ordinary search for similar instances to the query point in the same bucket can easily take O (log N) operations, where N is the number instances within the same bucket, but we reduced this to O (1) by using a pair of conventional hash functions. In the beginning, for mapping the d-dimensional quantized projection into a single linear index, we will use a conventional hash which is computed as following:

$$H_1 = \left(\sum_j w_j d_j\right) mod\ N_1 \tag{3}$$

where W_i are weights and N_1 is the hash table size. The main idea for using hash tables is to make sure that unrelated instances in the d-dimensional space are going to be allocated to different N_1 table entries for avoiding ''collisions'' in the case of which distinct instances hash to the same value. While a well-constructed hash will ensure a uniform distribution of entries, the risk of distinct instances colliding obviously increases, as the table gets smaller. To solve this problem, we will create a second hash H_2 of d-dimensional instances similar to H1 but with different weights and size [48]. Then, H_2 will be used to verify that instances retrieved from the hash table are near neighbors to our query. H_2 values (we will call it our reference instances) will be stored in the buckets selected by H1, and then on retrieval, we can compare our reference instances with instances retrieved from the matching buckets to identify the true matches, if they exist. Since these reference instances are short (for example, 13-bit values), their comparison will be much faster than comparing the entire d-dimensional original dataset (which requires a very expensive memory and computation requirements). Furthermore, the concurrent collisions probabilities under both H_1 and H_2 can be hugely reduced, even for a relatively small hash table.

For two reasons, we will use Locality sensitive Hashing in our proposed framework; the first one, is for creating subsets of the big data set by hashing instances into buckets, hence, the search for the global optimum in each bucket will be much easier than searching in whole data set at once. The second one, since we are working on big data, by using the LSH we will achieve very huge saving in terms of the framework running time, due to the fact, that is the complexity of LSH is O (d log n) vs. O (d n) for linear search techniques such as (K-NN), where d is the number of dimensions and n is the number of instances.

C. RECURRENT NEURAL NETWORKS

Schuster and Paliwa produced Bidirectional Recurrent Neural Networks (BRNN) in 1997 [49]. The basic idea of BRNNs is connecting hidden layers from opposite directions to the output. As a result of this structure, information from the past and future are available to the output at any time. Recurrent neural networks [50] can be considered as an extension of the ordinary feed forward multilayer perceptron networks, where inputs and outputs are vector of values instead of discrete values. Let us consider the input to a recurrent neural network by A = $\{a_t\}$ where $a_t \epsilon V^N$ is an input vector for each time step t. Moreover, let us consider the output as C = $\{c_t\}$ where $c_t \epsilon V^M$ is the output vector for each time step t. Our aim is to Model this distribution Q (C|A). Although, RNNs can be used for mapping the input vectors to output vectors, also, it can be used to predict the next input, which is called unsupervised manner by setting C = $\{c_t = a_t + 1\}$.

1) UNIDIRECTIONAL RECURRENT NEURAL NETWORKS

The output of a unidirectional recurrent neural network c_t can be calculated by:

$$Q\left(c_t \mid \{a_i\}_{i=1}^{t}\right) = \sigma(W_c h_t + b_c) \tag{4}$$

where

$$h_t = tanh\left(W_h h_{t-1} + W_a a_t + b_h\right) \tag{5}$$

W_c are the weights that connects the hidden layer to output layer, W_h are the weights that connects the hidden layer to hidden layer, and W_a are the weights that connects the input layer to the hidden layer. b_c are the biases of the output layer, and b_h are the biases of the hidden layer. For the final nonlinearity σ we can use sigmoid, tanh, and Relu as an activation function [51]. Due to this structure, the RNN will calculate

the output c_t based on the transmitted information through the hidden layers regardless of whether it depends directly or indirectly on the values $\{a_i\}_{i=1}^{t} = \{a_1, \ldots\ldots.., a_t\}$.

2) BIDIRECTIONAL RECURRENT NEURAL NETWORKS

Bidirectional Recurrent Neural Networks (BRNN) can be considered as an extension of the unidirectional recurrent neural networks by adding a second hidden layer, where the connections between the hidden to hidden layers are in the opposite direction [52]. Consequently, this model can exploit data from both directions, the past and the future. The output c_t can be calculated by:

$$Q\left(c_t \mid \{a_i\}_{i\neq t}\right) = \sigma(W_c^f h_t^f + W_c^b h_t^b + b_c) \tag{6}$$

where

$$h_t^f = tanh(W_h^f h_{t-1}^f + W_x^f x_t + b_h^f) \tag{7}$$

$$h_t^b = tanh(W_h^b h_{t+1}^b + W_a^b a_t + b_h^b) \tag{8}$$

If we compared the BRNN with the regular RNN, we can notice that the forward direction is designated by the superscript f, have separate non tied weights and activation functions from backward direction, which is designated by the superscript b. please mind that, in the proposed structure, c_t does not get data from a_t due to the none cyclic connections. Accordingly, this model can be used in an unsupervised manner by setting C = A, for predicting one-time step given all other time steps in the input sequence. Please mind that, in the backward pass of the BRNN, there should be two stages for the BPTT (Back Propagation Through Time) which are responsible for minimizing the MSE by adjusting the weights [53], since this job will be achieved by using the Grey wolf optimizer, we did not implement those two stages in our proposed algorithm.

3) BRNN PARAMETER OPTIMIZATION

Weights in a BRNN have two main roles; the first one is, deciding how much the output is affected by the input [54], and the second one is, controlling the learning rate of the hidden layers. Exactly as slope in linear regression, where the output is calculated by multiplying the weights to the inputs then added up. Weights are numerical values that control how much neurons are affecting each other. For any neuron, if the inputs are a_1, a_2, and a_3, and weights applied to them are w_1, w_2, and w_3. The output is:

$$c = f\ (a) = \sum_{j=1}^{n} a_j w_j \tag{9}$$

where n is the number of inputs. Generally, the weighted sum can be calculated by performing this array multiplication. Bias is an additional variable that can be used to adjusting the output along with the weighted sum of the inputs to the neuron [55]. The final output of a neuron is:

$$c = f\ (a) = \sum_{j=1}^{n} a_j w_j + b \tag{10}$$

where b is the bias.

4) BRNN UNSTABLE GRADIENT PROBLE

Most of deep neural networks, including bidirectional recurrent neural networks are suffering from a serious issue known as unstable gradient problem. This is due to, if we moved backward through the hidden layers, the gradient will start getting smaller. Which accordingly means that, the learning rate for neurons in the later layers is much faster than the learning rate for neurons in earlier layers. This issue is known as the vanishing gradient problem [56].

a: VANISHING GRADIENT PROBLEM

As we can notice from eq. 6, this formula is composed from terms multiplication in the form of $\sigma(w_c.h_t)$ except the last term (the bias). For understanding the behavior of each term in this formula, let's have a look at the plot of the activation function σ:

As we can notice from Fig.1, the peak value for the derivative is 0.25 at σ (0). The default initialization method for neural networks is as follows; the weights and biases will be chosen by using independent Gaussian distribution, with mean equal to 0 and standard deviation equal to 1. Consequently, all weights generally will satisfy $|w_c|$ <1. From previous conditions, we can assure that all $\sigma(w_c.h_t)$ terms will usually satisfy $|\sigma(w_c.\ h_t)|$ <0.25. Later on, this term will be multiplied by many similar terms, and the result of this product will start decreasing exponentially. As more terms are being added to the product, the gradient will start vanishing [57].

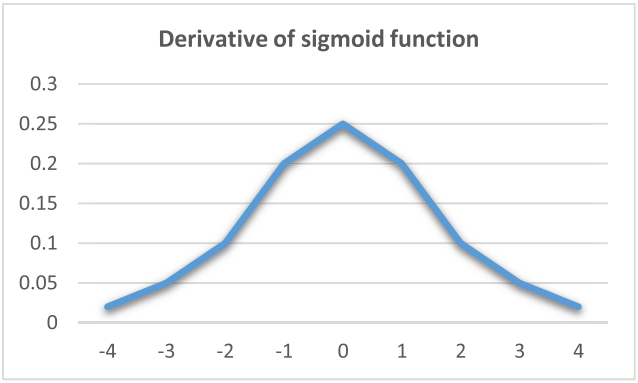

FIGURE 1. Derivative of sigmoid function.

b: EXPLODING GRADIENT PROBLEM

On the other hand, if we increased the weights w_c during training instead of decreasing, there could be a chance that the terms $\sigma(w_c.h_t)$ will no longer satisfy $|\sigma(w_c.h_t)|$ <0.25. In this case, if the terms $\sigma(w_c.h_t)$ increased enough, and became more than 1, then we will no longer have a vanishing gradient problem. In contrast, the gradient will start growing exponentially as we move backward through the layers. Instead of the vanishing gradient problem, we will have an exploding gradient problem.

To dominate the bidirectional recurrent neural network unstable gradient problem, the grey wolf optimizer will be

used for finding the optimal values of the weights and biases with only one condition, the values should be adjacent to 1 ($1 =< F^* =< 5$). Neither, very small, hence the gradient will vanish after the multiplication, nor very big, hence the gradient will explode after the multiplication.

D. GREY WOLF OPTIMIZER (GWO)

Grey wolves are one of apex predators, apex predators are at the top of the food chain. Grey wolves usually prefer to live in groups. The group size varies from 5–12 on average. The most interesting thing about grey wolves is that they have a very strict social leading hierarchy. The leaders called alphas, are a male and a female. The alpha is responsible for making important decisions to the group such as hunting, sleeping place, and so on. The whole group should obey the alpha's orders [31]. On the other hand, it had been observed that sometimes the alpha follows the other wolves in the group, which can be considered as some kind of democratic behavior. The alpha wolf is also called the leader wolf since the whole group should follow his/her orders. Interestingly, it is not necessarily for the alpha to be the strongest member among the group, but the best in terms of leading the group. Which implies that the discipline and organization of the group is more important than its strength [31].

Beta is the second level in the hierarchy of grey wolves. The betas are inferior wolves that are responsible for helping the alpha in making the right decisions or other group activities. The betas should respect the alpha, but on the other hand, they are also commanding the other lower-level wolves. They are playing dual role at the same time, an advisor to the alpha and discipliner for the group. The betas are the heavy hand of the alpha, they are responsible for reinforcing the alpha's commands throughout the group and reporting feedback to the alpha. Omega is the lowest ranking grey wolf. The omega plays the role of scapegoat [31]. Omega wolves always have to succumb to all other dominant wolves. Additionally, they are the last wolves that are allowed to eat. Hence, the omegas may be seen as not an important individual in the group, but it has been observed that in case of losing the omegas, the whole group may face internal fighting and problems. This is because the omegas are venting of violence and frustration of all wolves in the group. Accordingly, the role of the omegas is to assist satisfying the entire group and maintaining the dominance structure.

If a wolf is not an alpha, beta, or omega, he/she is called inferior (or according to some references delta). Delta wolves have to succumb to alphas and betas, but they command the omega. Hunters, sentinels, scouts, and elders belong to this category. Hunters are responsible for helping the alphas and betas when hunting prey and providing food for the group. Sentinels are responsible for guaranteeing and protecting the safety of the group. Scouts job is watching the boundaries of the territory and warning the group in case of any danger. The experienced wolves who have been alpha or beta are the Elders of the group. The social hierarchy of grey wolves is interesting, but its group hunting technique is even more interesting. The main stages of grey wolf hunting according to [58] are as follows:

1) Tracking, chasing, and approaching the prey.
2) harassing, and encircling the prey until it stops moving.
3) Attack towards the prey.

In this section, the GWO will mathematically modeling the grey wolves hunting technique (tracking, encircling, and attacking prey) and the social hierarchy for performing optimization.

1) SOCIAL HIERARCHY

When we are designing GWO, in order to mathematically model the social hierarchy of wolves, we will consider the best solution as the alpha (α). Accordingly, the second-best solution will be named beta (β), and the third best solution will be named delta (δ). Then, the rest of the best solutions will be assumed as omegas (ω). In the GWO algorithm the hunting technique (optimization) is controlled by α, β, and δ. The ω wolves are following these three wolves [31].

2) ENCIRCLING PREY

As we said before, the grey wolves are encircling prey during the hunt. For modelling encircling behavior mathematically, the following equations are proposed:

$$\vec{D} = \left|\vec{C}.\vec{F}_p\,(t) - \vec{F}(t)\right| \tag{11}$$

$$\vec{F}\,(t+1) = \vec{F}_p\,(t) - \vec{A}.\vec{D} \tag{12}$$

where t represents the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors, which are calculated as in Eq. (14) and Eq. (15), $\vec{F}_p$ indicates the position vector of the prey, and $\vec{F}$ is the grey wolf's position vector. In each iteration, $\vec{F}$ will be updated if there is a better solution.

$$\vec{a} = 2 - t(\frac{2}{Max_{iter}}) \tag{13}$$

$$\vec{A} = 2\vec{a}.\vec{r}_1 - \vec{a} \tag{14}$$

$$\vec{C} = 2.\vec{r}_2 \tag{15}$$

where t is the loop counter, Max_{iter} is the maximum number of iterations in this loop, $\vec{r}_1$, $\vec{r}_2$ are random vectors in [0,1], and $\vec{a}$ is linearly decreased from 2 to 0 over the course of iterations. For testing the effects of Eqs. (11) and (12), assume a two-dimensional position vector and some of the possible neighbors as shown in Fig. 2. As we can see in this figure, a grey wolf in the position (X, Y) can update its position according to the position of the prey (X^*, Y^*). By adjusting the value of $\vec{A}$ and $\vec{C}$ vectors with respect to the current position, different places around the best agent can be reached. For example, (X^*–X, Y^*) can be reached by setting $\vec{A} = (1,0)$ and $\vec{C} = (1,1)$. The random vectors r1 and r2 allow grey wolves to reach any position between the points shown in Fig. 2. Accordingly, by using Eqs. (11) and (12) a grey wolf can update its position randomly inside the space

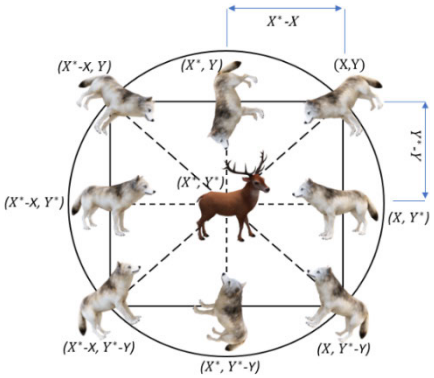

FIGURE 2. Grey wolves' positions and their possible next locations.

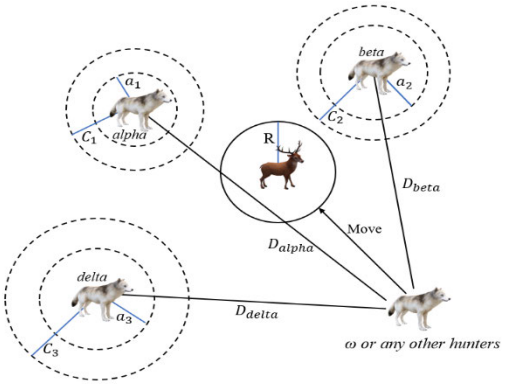

FIGURE 3. search agents' positions updating in GWO.

around the prey. The same approach can be applied to an n-dimensions search space, and the grey wolves will move in (hyper-spheres/cubes) around the best solution obtained so far [31].

3) HUNTING

One of the grey wolves' abilities is determining the location of prey and encircling them. The alpha usually guides the whole hunt operation. Sometimes, the beta and delta are participating in hunting operation. But, in our virtual 2D search space, we do not have any idea about the location of the prey (optimum). For mathematically simulating the hunting behavior of grey wolves, we will assume that the alpha, beta, and delta knows the potential location of prey. So, we will save the best three results obtained so far, and then, the other grey wolves (including the omegas) will be forced to update their positions according to the position of the best three search agents [31]. For achieving this, we proposed the following formulas:

$$\vec{D_\alpha} = \left|\vec{C_1} * \vec{F_\alpha} - \vec{F}\right|, \quad \vec{D_\beta} = \left|\vec{C_2} * \vec{F_\beta} - \vec{F}\right|, \quad \vec{D_\delta} = \left|\vec{C_3} * \vec{F_\delta} - \vec{F}\right| \tag{16}$$

$$\vec{F_1} = \vec{F_\alpha} - \vec{A_1} * \left(\vec{D_\alpha}\right), \quad \vec{F_2} = \vec{F_\beta} - \vec{A_2} * \left(\vec{D_\beta}\right), \quad \vec{F_3} = \vec{F_\delta} - \vec{A_3} * (\vec{D_\delta}) \tag{17}$$

$$\vec{F}\,(t+1) = \frac{\vec{F_1} + \vec{F_2} + \vec{F_3}}{3} \tag{18}$$

Fig. 3 shows in a 2D search space, how a grey wolf (search agent) updates its position according to alpha, beta, and delta. Hence, as we can notice from Fig. 3, the final position of a grey wolf (search agent) will be in a random place within a circle in the search space defined by the positions of alpha, beta, and delta. In other words, the position of the prey is estimated by alpha, beta, and delta, and other wolves are following this estimation and updating their positions randomly around the prey [31].

4) ATTACKING PREY (EXPLOITATION)

As we said above, the last stage in the hunt is attacking the prey when it stops moving. For modeling approaching the prey mathematically, we will decrease the value of $\vec{a}$. Please note that, the range of $\vec{A}$ will be also decreased by $\vec{a}$. Which means, $\vec{A}$ is a random value in the interval $[-2a, 2a]$, where a is decreasing from 2 to 0 over the course of iterations. When random values of $\vec{A}$ are in $[-1, 1]$, the next position of a grey wolf (search agent) can be in any position between its current position and the position of the prey. Fig. 4(a) shows that by setting $|A| < 1$ it forces the wolves to attack towards the prey. With the operators proposed so far, the GWO algorithm allows its search agents to update their position based on the location of the alpha, beta, and delta, and attack towards the prey [31]. But, with these parameters only, the GWO algorithm will probably suffer from stagnation in local solutions. Although, the proposed encircling mechanism shows exploration to some extent, but GWO algorithm still needs more parameters for emphasizing exploration.

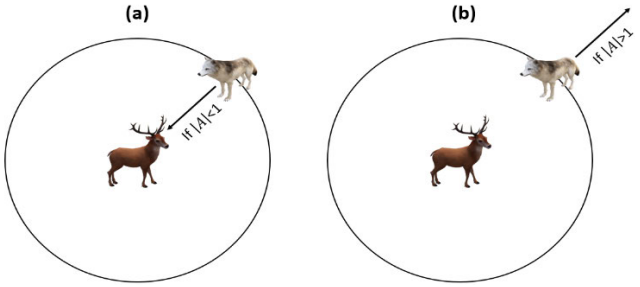

FIGURE 4. GWO Hunting techniques: (a) attacking the prey (convergence), (b) searching for prey (divergence).

5) SEARCH FOR PREY (EXPLORATION)

As mentioned before, grey wolves are searching for the prey according to the position of the alpha, beta, and delta. The typical procedure includes two steps, divergence from each other searching for prey, and convergence for attacking the prey. For modeling the divergence mathematically, $\vec{A}$ will be utilized with random values greater than 1 or less than -1 for forcing the grey wolf (search agent) to diverge from the prey. This technique is emphasizing the exploration and allowing the GWO algorithm to search globally. From Fig. 4(b) we can see that $|A| > 1$ is forcing the grey wolves (search agents) to diverge from the prey hoping to find a better prey.

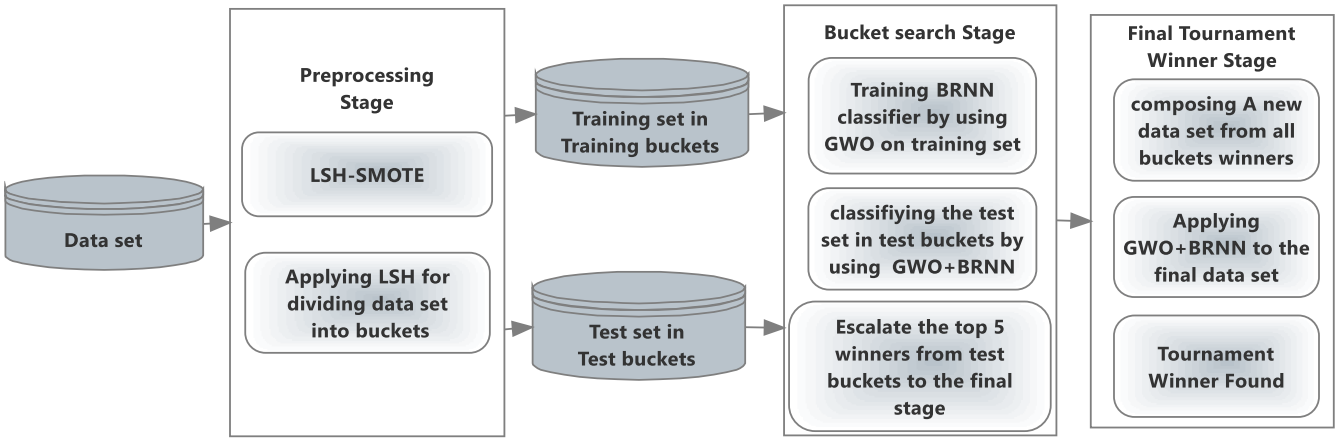

FIGURE 5. The block diagram for our proposed LSHGWOBRNN framework.

$\vec{C}$ is Another component of GWO that emphasizes exploration, as we can see in Eq. (15), the $\vec{C}$ vector contains random values in [0, 2]. This parameter is responsible for providing the prey with random weights for emphasizing ($C > 1$) or deemphasizing ($C < 1$) the effect of prey randomly in defining the distance in Eq. (11). Accordingly, this will increase the exploration, increases the GWO random behavior throughout optimization, and will increase the local optima avoidance. In contrast to A, C here will not linearly decrease. For emphasizing the exploration not only during initial iterations, but also during final iterations, C vector is intentionally required to provide random values at all times [31]. This parameter is very important for local optima avoidance, particularly in the final iterations.

Another important feature of the C vector, that it can simulate the effect of obstacles that prevents grey wolves from easily approaching prey in nature. In fact, the obstacles in nature may appear in the hunting paths of grey wolves and preventing them from quickly approaching prey, and this is exactly what the vector C does. Depending on the position of the grey wolf, it can give the prey a random weight and make it farther and harder for wolves to reach it, or vice versa. To summarize, the search process in the GWO algorithm begins with creating a random population of grey wolves (search agents). Over the course of iterations, the possible location of the prey is estimated by alpha, beta, and delta wolves. Each search agent is updating its distance from the prey. For emphasizing exploration and exploitation respectively, the parameter a is decreased from 2 to 0. Search agents are diverging from the prey when $\left|\vec{A}\right| > 1$, and converging towards the prey when $\left|\vec{A}\right| < 1$. At the end, the GWO algorithm is terminated when an end criterion is reached.

III. COMPONENT OF THE PROPOSED FRAMEWORK

The main idea for our proposed framework is based on the tournament idea. As in the tournaments there are the groups level, then the best candidates will be escalated to the next level of the tournament, tell the final level of the tournament. In our framework we will hash the data set into buckets, then, we will use the GWO+BRNN for finding the top 5 best candidates in each bucket and escalate them to the next level in our tournament, which is similar exactly to the group idea. The next level of our tournament is the final tournament level, in this level we will compose a new data set from all buckets winners (the top 5 best winners from each bucket), then, by using the GWO+BRNN we will choose the best of the best to be our tournament winner, which will be accordingly, the global optimum of the whole data set (the instance with the lowest MSE). The original components of the proposed framework are shown in Fig.5. It consists of three stages preprocessing stage, bucket search stage, and final tournament winner stage.

A. PREPROCESSING STAGE

In this stage, we will use LSH-SMOTE algorithm for resolving the imbalanced classes problem as it has provided a very competitive results compared to SMOTE algorithm and 9 most recent variations of SMOTE algorithm [59], then, we will apply the LSH algorithm for hashing the data set into buckets for simplifying the search for global optimum in each bucket.

1) LOCALITY SENSITIVE HASHING SYNTHETIC MINORITY OVERSAMPLING TECHNIQUE (LSH-SMOTE)

In this algorithm, we will use the LSH-SMOTE algorithm for solving the imbalanced data set classes quickly and efficiently. This algorithm has been tested against SMOTE algorithm and nine most recent variations of SMOTE algorithm and proved that, it is the best among them in terms of running time and classification results [59]. This algorithm is based on the hashing idea and creating buckets, and assigning all items with similar hash code to the same bucket, which is accordingly will increase the probability of collision between similar items. Later on, we will select the most colliding items from each bucket, and use the Euclidean distance for sorting them and select only the top five among them. At the end, a list that contains all the top five colliding items from each

Method1: Locality Sensitive Hashing (LSH) algorithm

```
Input: dataset
Load dataset
Dividing the dataset into training set (80%) and test set (20%)
k = Hash key length
T = The Number of hash tables
Ntr= the number of instances in the training set
Nts= the number of instances in the test set
LSHtrdata= training set hashed into buckets
LSHtsdata=test set hashed into buckets
Bucketrtdata= the resulted training set in one bucket
Bucketstdata= the resulted test set in one bucket
NBtr=number of training instances in a training bucket
NBts=number of test instances in a test bucket
numtrBuck= number of training buckets
numtsBuck= number of test buckets
Applying LSH-SMOTE algorithm for resolving the imbalanced classes problem
for i =1 to ntr
LSHtrdata (i) ← LSH (training set, k, T)          # Apply LSH procedure to training set
numtrBuck ← length (LSHtrdata.Index(i))           # Number of training buckets created
end for
for j =1 to numtrBuck
Assigning instances with the same hash code to the same bucket
NBtr(j) ← [bucketrtdata.Index(j)]
end for
for j =1 to nts
LSHtsdata (j) ← LSH (test set, k, T)              # Apply LSH procedure to test set
numtsBuck ← length (LSHtsdata.Index(j))           # Number of test buckets created
end for
for j =1 to numtsBuck
Assigning test instances with the same hash code to the same bucket
NBts(j) ← [bucketstdata.Index(j)]
end for
Output: LSHtrdata, numtrBuck, NBtr, LSHtsdata, NBts, numtsBuck
```

FIGURE 6. Locality sensitive hashing algorithm pseudo code.

bucket is sent back to the main SMOTE class for creating synthetic samples.

2) APPLYING LOCALITY SENSITIVE HASHING (LSH) FOR HASHING THE DATA SET INTO BUCKETS

Since the main reason for local optima is the big size of data set and its multi-dimensional space, which makes finding a global optimum in particular area not necessarily the global optimum for the whole data set, which leads to the local optima problem. Therefore, in this phase we will apply the LSH algorithm for hashing the data set into buckets. The main goal of this phase is creating smaller subsets of the data set, which accordingly will ease the search for global optimum in each bucket alone. Hence, we can find the global optimum for the whole data set more easily and precisely. The pseudo code for the LSH algorithm is provided in Fig. 6.

B. BUCKET SEARCH STAGE

In this stage we will describe the details of our proposed algorithm (GWO+BRNN) for adjusting the weights and biases of the bidirectional recurrent neural network. Then, it will be used for finding the best 5 candidates (search agents) in each bucket, and escalate them to the final tournament stage.

1) GWO PARAMETERS INITIALIZATION

In this step, we will initialize the GWO parameters as follows: we will initialize the number of wolves (search agents) to 100, we will initialize the number of iterations to 50. Those values were selected after performing several experiments, when we tried to increase the number of wolves and iterations more than that, we did not achieve any improvement in the results (AUC & MSE), in contrast, it increased the algorithm overhead and complexity.

2) GWO-BASED BRNN TRAINER

As we stated before, the BRNN training is strongly affected by the values of weights and biases. The trainer job is reaching the highest classification results by selecting the optimal values for weights and biases [60]. The variables provided to the GWO algorithm should be in vector form as follows:

$$\vec{V} = \left\{\vec{W}, \vec{b}\right\} = \{W_{11}, W_{12}, \ldots, W_{mm}, h, b_1, b_2, \ldots, b_m \quad (19)$$

where the number of the input nodes is m, the connection weight from the x_{th} node to the y_{th} node is W_{xy}, and the bias is b_y.

After defining the GWO parameters, our next step is defining the objective function for the GWO algorithm. The Mean Square Error (MSE) is one of the common metrics for evaluating the BRNN performance [61]. This metric calculates the

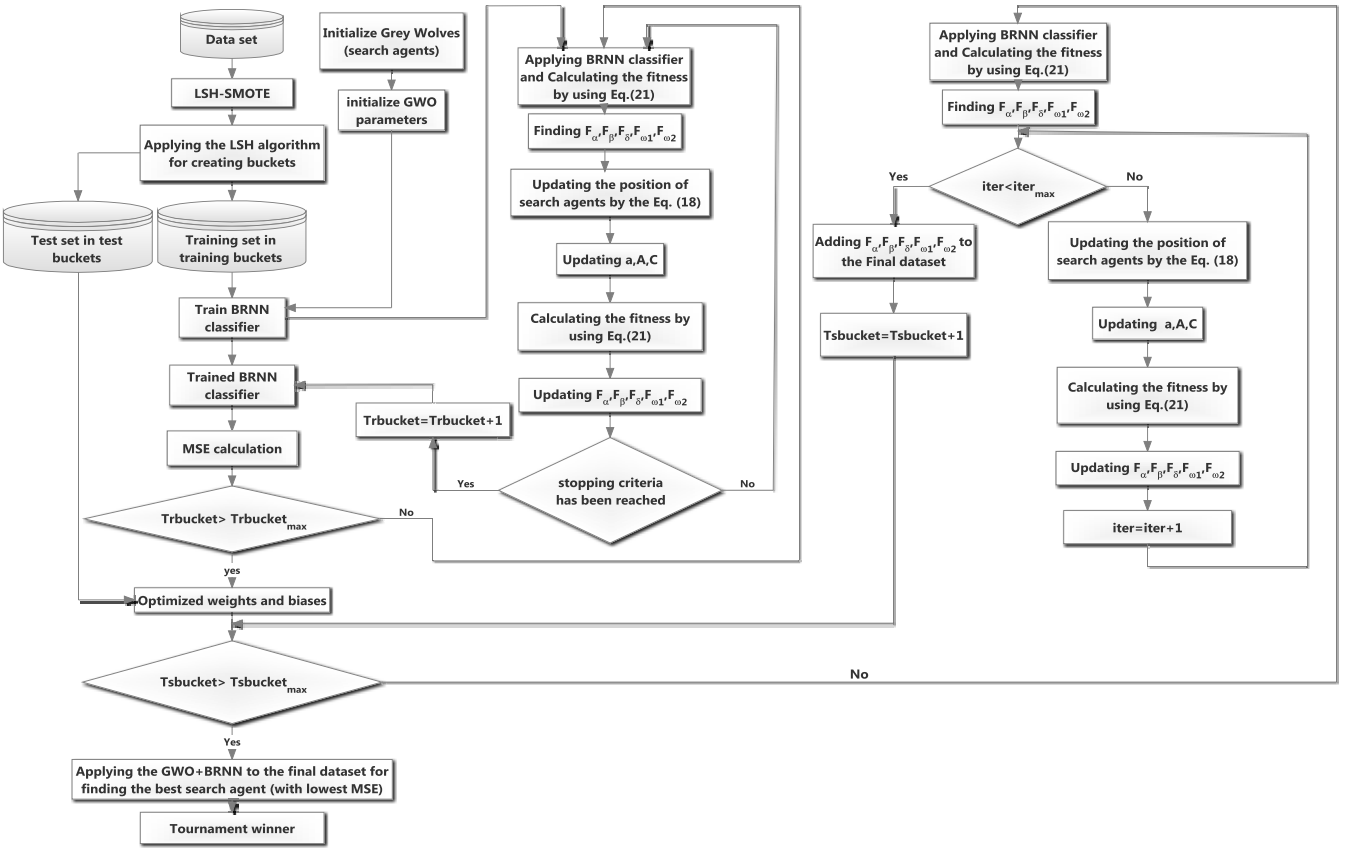

FIGURE 7. Flowchart of the proposed LSHGWOBRNN algorithm.

difference between the required output and the actual output of the BRNN by applying a set of training instances to the BRNN according to the following equation:

$$MSE = \sum_{x=1}^{n} (o_x^h - d_x^h)^2 \tag{20}$$

where n is the number of outputs, d_x^h is the optimal output of the x_{th} input neuron when the h_{th} training instance is used, and o_x^h is the actual output of the x_{th} input neuron when the h_{th} training instance appears in the input. For increasing the efficiency of BRNN, the performance of the BRNN should be calculated according to the average of MSE over all the training instances as follows:

$$\overline{MSE} = \sum_{h=1}^{z} \frac{\sum_{x=1}^{n} (o_x^h - d_x^h)^2}{z} \tag{21}$$

where z is the number of training instances. Finally, the problem of training BRNN can be formulated with the variable and average MSE for the GWO algorithm as follows:

$$Minimize : f(\vec{v}) = \overline{MSE} \tag{22}$$

Fig. 7 depicts the flowchart for our proposed algorithm LSHGWOBRNN. At the beginning, we applied the LSH-SMOTE algorithm for the imbalanced classes problem, then we applied the LSH algorithm for hashing the data set into buckets. After that, for training the BRNN we used the GWO algorithm for optimizing the weights and biases of the BRNN. Later on, we used the trained GWO+BRNN for searching for the best five instances in each bucket and composed a new data set called final data set. Finally, we used the GWO+BRNN for searching for the tournament winner (the best instance with lowest MSE).

3) TERMINATION POINT

The GWO search process will be terminated in one of two cases, either if the maximum number of iterations has been reached, or the best solution has not been changed over four iterations. The GWO algorithm pseudo code is shown in figure 8, and Fig. 9 is depicting the BRNN algorithm pseudo code, while Fig. 10 is depicting our proposed GWO+BRNN algorithm pseudo code within the bucket search stage.

Fig. 8 is illustrating the grey wolf optimizer pseudo code, this is a modified version of the one proposed by Seyed Ali Mirjalili in [31], in three parts, the first one, for solving the unstable gradient problem we restricted the search domain to (1 =<F* =<5), the second one, if the best solution F_α has not been changed for four iterations, the search process will be terminated without reaching the maximum number of iterations and F_α will considered as the best solution. And the third one, is that the original GWO reports only 4 best search agents, but in our modified one, we will report the best 5 search agents F_α, F_β, F_δ , $F_{\omega 1}$, and $F_{\omega 2}$.

Fig. 9 is illustrating the bidirectional recurrent neural network (BRNN) algorithm pseudo code, this algorithm involves

Method2: Grey Wolf Optimization Algorithm (GWO modified)

1. F_α=*the best wolf = F**
2. F_β= *second best wolf*
3. F_δ= *third best wolf*
4. $F_{\omega 1}$= *fourth best wolf*
5. $F_{\omega 2}$= *fifth best wolf*
6. *nrc=No result Change*
7. *nrc =0*
8. *Max-iter= maximum number of iterations*
9. *Initialize the Wolves population Fi (i = 1, 2, ..., n)*
10. *Initialize a, A, and C*
11. *Calculate the fitness function for each wolf by using Eq.(21)*
12. ***while*** *(t < Max-iter and nrc<4)*
13. ***for*** *each wolf*
    Update the position of the current wolf by the Eq.(18)
14. ***end for***
15. *Update a, A, and C*
16. *Calculate the fitness function for all wolves by using Eq.(21)*
17. *Update* F_α, F_β , F_δ , $F_{\omega 1}$,*and* $F_{\omega 2}$
18. *t=t+1*
19. ***if*** $(F^*_{crnt} = F^*_{prev})$
20. *nrc=nrc+1*
21. ***end if***
22. ***end while***
23. *Return* F_α, F_β , F_δ , $F_{\omega 1}$,*and* $F_{\omega 2}$

FIGURE 8. Grey wolf optimization algorithm pseudo code

Method3: Bidirectional Recurrent Neural Network (BRNN)Algorithm

1. *In= the number of input layers*
2. *Hn= the number of hidden layers*
3. *On= the number of output layers*
4. *n= number of data set instances*

Forward pass

5. *for x=1 to Hn*
6. *for y=1 to n*
   calculating the forward pass for the forward hidden layer's activation function h_t^f *using eq. (7)*
7. *end for*
8. *for y=n to 1*
9. *calculating the backward pass for the backward hidden layer's activation function* h_t^b *using eq. (8)*
10. *end for*
11. *end for*
12. *for x=1 to On*
13. *calculating the forward pass for the output layer using the previous stored activations using eq. (6)*
14. *end for*

Backward pass

15. *for x= On to 1*
16. *calculating the backward pass for the output layer using the previous stored activations using eq. (6)*
17. *end for*
18. *for x=1 to Hn*
19. *for y=1 to n*
    calculating the backward pass for the forward hidden layer's activation function h_t^f *using eq. (8)*
20. *end for*
21. *for y=n to 1*
22. *calculating the forward pass for the backward hidden layer's activation function* h_t^b *using eq. (7)*
23. *end for*
24. *end for*

FIGURE 9. Bidirectional recurrent neural network pseudo code.

of two stages, the forward pass and the backward pass. In the forward pass stage, we will pass all inputs through the BRNN for determining all the predicted outputs. And this can be carried out through two phases. In the first phase, a forward pass will be done for forward states (from y = 1 to y = n) and backward states (from y = n to y = 1). In the second phase, a forward pass will be done for output neurons. In the backward pass, the derivative of the objective function σ will be calculated using by two phases. In the first phase, a backward pass will be done for output neurons. In the second phase, a backward pass will be done for backward states (from y = 1 to y = n) and forward states (from y = n to y = 1).

Algorithm1: Bucket search stage

```
Input: LSHtrdata, NBtr, numtrBuck, LSHtsdata, NBts, numtsBuck
nr=the number of training instances in training buckets
ns= the number of test instances in test buckets
In= the number of input layers
Hn= the number of hidden layers
On= the number of output layers
Nf= the number of features in the data set
dim= data set dimensions
LSHtrdata= training set hashed into buckets
LSHtsdata=test set hashed into buckets
numtrBuck= number of training buckets
numtsBuck= number of test buckets
NBtr=number of training instances in a training bucket
NBts=number of test instances in a test bucket
nn= the number of instances in the final dataset
Finaldataset= the final data set that contains the best 5 instances (lowest MSE) from each bucket
(Calling method1) LSH algorithm
Dim = (nf +1) *Hn+1                                   # Calculating the dimensions of the data set
for ww=1 to nf*Hn
W(ww)= (calling method2) GWO (1, ww)                  # finding optimal weights by using GWO
End for
for bb= nf *Hn+1 to dim
B (bb-( nf *Hn)) =GWO (1, bb)                         # finding optimal biases by using GWO
end for
fitness=0;
for j=1 to numtrBuck
for i=1 to NBtr(j)
actual value(i)= (calling method3) GWO+BRNN (In, Hn, On, W, B, LSHtrdata)  # the classifier is being trained on training set
fitness=fitness+( Actual value(i)-Optimal value(i))^2                      # calculating MSE (Mean Square Error)
end for
end for
nn=0
fitness=0;
for j=1 to numtsBuck
for i=1 to NBts(j)
Actual value(i)= (calling method3) GWO+BRNN (In, Hn, On, W, B, LSHtsdata)  # test set data classification
fitness=fitness+( Actual value(i)-Optimal value(i))^2                      # MSE calculation
end for
return F_α, F_β , F_δ , F_ω1 ,and F_ω2
adding the best top 5 search agents (wolves) to the Finaldataset
nn=nn+5
end for
Output: Finaldataset, nn
```

FIGURE 10. Bucket search stage pseudo code.

Fig. 10 illustrates the bucket search stage pseudo code, firstly, we will apply the LSH-SMOTE algorithm for solving the imbalanced classes problem. Secondly, we will apply the LSH algorithm for hashing the data set instances into buckets. Thirdly, the GWO algorithm will be used for providing the BRNN classifier with weights and biases. Fourthly, our BRNN classifier will be trained on the training set and the MSE for the training phase will be calculated. Fifthly, our GWO+BRNN classifier will be applied to the test set for classification and the MSE for the classification phase will be calculated. Finally, a new data set called final data set is composed from all bucket's winners (top 5 instances with the lowest MSE).

C. FINAL TOURNAMENT WINNER STAGE

In this stage we will search for the final tournament winner (the best instance with the lowest MSE), among all winners from all buckets in the new composed data set named Final dataset. First of all, we will calculate the dimensions of the new data set. Then, the GWO algorithm will be used for providing the BRNN classifier with weights and biases. Finally, we will apply the GWO+BRNN classifier for classifying the data set and calculating the MSE for all instances in this dataset for determining the tournament winner (the best instances with the lowest MSE). At the end, calculating the classification rate (accuracy), and AUC for the tournament winner. The algorithm for this stage is illustrated in Fig. 11.

IV. EXPERIMENTAL RESULTS

Our experiment has been performed by using (i7-8700K 4.3 GHz, 16GB RAM, 1TB SSD, Windows 10 pro) with Weka 3.9.1 and Matlab R2018a. Our experiment will include two main sections. In the first section, our proposed framework LSHGWOBRNN will be tested against seven classifiers (Naive Bayes, AdaBoostM1, Decision Table, and

Algorithm2: Final tournament winner stage

```
Input: Finaldataset, nn
In= Number of input layers
Hn= Number of hidden layers
On= Number of output layers
Nf= the number of features in the data set
dim= data set dimensions
nn= the number of instances in the final dataset
Finaldataset= the final data set that contains the best 5 instances (lowest MSE) from each bucket
Dim = (nf +1) *Hn+1                                   # Calculating the dimensions of the data set
for ww=1 to nf*Hn
W(ww)= (calling method2) GWO (1, ww)                  # finding optimal weights by using GWO
End for
for bb= nf *Hn+1 to dim
B (bb-( nf *Hn)) =GWO (1, bb)                         # finding optimal biases by using GWO
end for
fitness=0;
for i=1 to nn
Actual value(i)= (calling method3) GWO+BRNN (In, Hn, On, W, B, LSHtsdata)   # data set classification
fitness=fitness+( Actual value(i)-Optimal value(i))^2                       # MSE calculation
end for
return F_α (the best instances with the lowest MSE)
calculate the classification rate and the AUC (Area Under Curve) for F_α (the tournament winner)
Output: F_α (MSE, classification rate, and AUC)
```

FIGURE 11. Final tournament winner stage pseudo code.

Random Tree), in addition to GWO+MLP, which is published in 2015 [62], WOA+MLP, and WOA+BRNN [71], will be performed over eight highly imbalanced data sets obtained from the KEEL Data Set Repository (Imbalance ratio higher than 9) [63], and one big dataset that has been used in ECBDL 14 Big Data Mining Competition 2014 [64]. In the second part, our proposed framework LSHGWOBRNN will be compared against GWO+MLP over all data sets in that paper [62]. In [62], Seyed Ali Mirjalili used the GWO for providing the neural network (multi-layer perceptrons) with optimal weights and biases.

A. PERFORMANCE EVALUATION METRICS

Many metrics have been proposed for the evaluation of imbalanced classification performance evaluation problems. Most of them are based on the 2 × 2 confusion matrix as depicted in Table 1.

TABLE 1. The confusion matrix.

	Predicted positive	Predicted negative
Positive class	True positive (TP)	False negative (FN)
Negative class	False positive (FP)	True negative (TN)

One of the most commonly used metrics for evaluating the performance of classification techniques is the overall accuracy [65], which can be calculated as following:

$$Acc = \frac{TP + TN}{TP + FN + TN + FP} \tag{23}$$

However, it has been proved by many researches that, in the context of imbalanced data sets the overall accuracy is not the optimal metric for such problems, since the results will be strongly biased toward the majority class [66]. Hence, finding other metrics that can efficiently measure the classification performance of imbalanced data sets is essential.

Two simple metrics that are proven to calculate the classification performance over imbalanced data sets efficiently are the true positive rate (or sensitivity or recall) and the true negative rate (or specificity) [67], which is the percentage of instances (positive and negative, respectively) are correctly classified [68]:

$$sensitivity = \frac{TP}{TP + FN} \tag{24}$$

$$specificity = \frac{TN}{TN + FP} \tag{25}$$

Another very important performance evaluation metric is the area under the ROC curve (AUC), which is the geometric mean of accuracies, the precision, the F-measure and the area under the precision-recall curve, among others [69]. Generally, for imbalanced data sets these are good indicators of classification performance because they are independent from the distribution of instances between classes. The AUC, which is one of the most commonly used metrics for evaluating imbalanced classes problem, will be the method used in this part of our experiment for performance evaluation. The AUC which can be defined by a single point on the ROC curve, is also referred to as balanced accuracy or macro-average [70], which can be calculated as follows:

$$AUC = \frac{sensitivity + specificity}{2} \tag{26}$$

Our benchmark will be carried out by performing two main experiments. In both experiments, our proposed framework LSHGWOBRNN will be tested against the seven classifiers

TABLE 2. Experiment datasets characteristics.

Data Set name	#Examples	#Attributes	IR
yeast-2_vs_4	514	8	9.08
yeast-1_vs_7	459	7	14.3
yeast-1-4-5-8_vs_7	693	8	22.1
yeast-2_vs_8	482	8	23.1
yeast4	1484	8	28.1
yeast-1-2-8-9_vs_7	947	8	30.57
yeast5	1484	8	32.73
yeast6	1484	8	41.4
ECBDL 14 Data set	2897917	23	58.58

TABLE 3. LSH-SMOTE parameters settings.

Algorithm	Parameters
LSH-SMOTE	Number of nearest neighbors (K=5), oversampling percentage = 50%, number of Hashes (H= 5), number of Hash Tables (T=4).

over nine data sets. The first experiment is done without preprocessing while the second one uses LSH-SMOTE algorithm for preprocessing the data sets. The four classifiers benchmarks in all experiments will be performed on Weka 3.9.1, only GWO+MLP, WOA+MLP, WOA+BRNN, and our proposed framework experiments will be performed on Matlab R2018a. Also, the preprocessing stage for LSH-SMOTE will be also performed on Weka 3.9.1, then the preprocessed data sets will be imported into Matlab for performing the GWO+MLP, WOA+MLP, WOA+BRNN, and our proposed framework benchmarks. The characteristics of data sets used are depicted in Table 2.

The original ECBDL'14 dataset has 32 million instances with 631 attributes distributed among two classes, the uncompressed dataset size is about 56GB of disk space. Our experiments have been performed by using test set only which is 5GB of disk space, and has 2897917 instances, and we have reduced the number of attributes from 631 to 23 for simplifying the computations.

For the solving the imbalanced classes problem, we will use LSH-SMOTE algorithm for preprocessing our data sets. The configuration for LSH-SMOTE algorithm is depicted in Table 3

B. FIRST EXPERIMENT

In this experiment, our proposed framework LSHGWO-BRNN will be tested against seven classifiers [71] over nine highly imbalanced data sets, over two sub experiments, without preprocessing, and with LSH-SMOTE preprocessing in terms of AUC and MSE (Local Optima Avoidance).

1) EXPERIMENT ONE (WITHOUT PREPROCESSING)

In this experiment, we will test our proposed framework LSHGWOBRNN against the seven classifiers (Naive Bayes, AdaBoostM1, Decision Table, Random Tree, GWO+MLP, WOA+MLP, and WOA+BRNN) over nine data sets without preprocessing in terms of AUC and MSE (Local Optima Avoidance). The AUC results will be depicted in Table 4, while the MSE results will be depicted in Table 5.

Table 4 and Fig. 12 shows the AUC results for all classifiers over nine data sets without preprocessing. For yeast-2_vs_4 dataset, our algorithm ranked first with score (0.989). For yeast-1_vs_7 dataset, our algorithm ranked second with score (0.831), while the first algorithm AdaBoost achieved (0.832). For yeast-1-4-5-8_vs_7 dataset, our algorithm ranked second with score (0.659), while the first algorithm Naive Bayes achieved (0.662). For yeast-2_vs_8 dataset, our algorithm ranked second with score (0.811), while the first algorithm Naive Bayes achieved (0.821). For yeast4 dataset, our algorithm ranked first with score (0.909). For yeast-1-2-8-9_vs_7 dataset, our algorithm ranked second with score (0.762), while the first algorithm WOA+MLP achieved (0.764). For yeast5 dataset, our algorithm ranked First with score (0.992). For yeast6 dataset, our algorithm ranked first with score (0.927). For ECBDL 14 dataset, our algorithm ranked first with score (0.731). Finally, we achieved the best average over all other algorithms (0.848), then, WOA+BRNN algorithm ranked second with score (0.832), then, AdaBoost algorithm ranked third with score (0.830), then, WOA+MLP algorithm ranked fourth with score (0.829),then, GWO+MLP algorithm ranked fifth with score (0.826), then, Naive Bayes

TABLE 4. AUC results without preprocessing.

data sets	Naive Bayes	AdaBoost	Decision Table	Random Tree	GWO+ MLP	WOA+ MLP	WOA+ BRNN	LSHGWOB RNN
yeast-2_vs_4	0.908	0.975	0.929	0.878	0.976	0.978	0.982	**0.989**
yeast-1_vs_7	0.801	**0.832**	0.546	0.579	0.798	0.798	0.811	0.831
yeast-1-4-5-8_vs_7	**0.662**	0.639	0.498	0.501	0.639	0.629	0.637	0.659
yeast-2_vs_8	**0.821**	0.800	0.682	0.613	0.793	0.799	0.801	0.811
yeast4	0.855	0.893	0.559	0.634	0.899	0.889	0.907	**0.909**
yeast-1-2-8-9_vs_7	0.756	0.723	0.542	0.629	0.734	**0.764**	0.741	0.762
yeast5	0.986	0.985	0.822	0.814	0.984	0.979	0.990	**0.992**
yeast6	0.923	0.911	0.539	0.720	0.904	0.895	0.910	**0.927**
ECBDL 14	0.684	0.713	0.557	0.583	0.707	0.728	0.705	**0.731**
Average	0.822	0.830	0.630	0.661	0.826	0.829	0.832	**0.848**

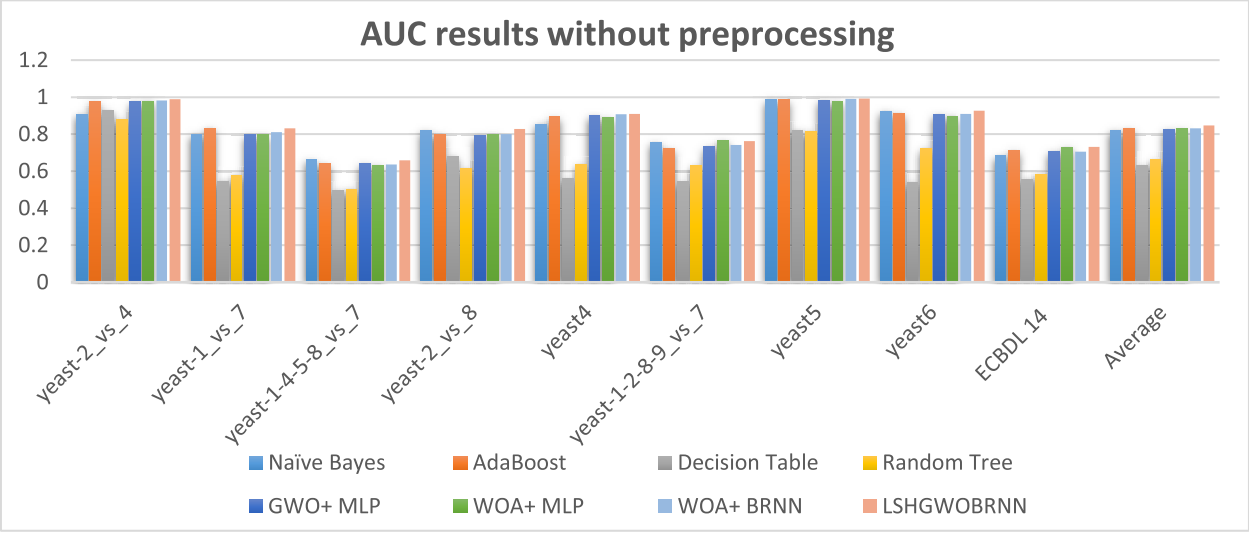

FIGURE 12. AUC results without preprocessing.

algorithm ranked sixth with score (0.822), then, Random Tree algorithm ranked seventh with score (0.661), and finally, Decision Table algorithm ranked eighth with score (0.630). From table 4 results, we can notice that our proposed framework LSHGWOBRNN did not perform very well in this stage (we achieved the highest score in four data sets only out of nine data sets), due to the fact, that all data sets in this stage were imbalanced and have not been preprocessed. In contrast, Naive Bayes algorithm alone outperformed our proposed framework LSHGWOBRNN in two data sets out of nine data sets. These bizarre results of Naive Bayes algorithm are expected due to two weaknesses of this classifier. The first issue, if the classes are imbalanced, the Naive Bayes chooses poor weights for the decision boundary. The second issue, Naive Bayes assumes that all features are independent. These weaknesses can be summarized as severe assumptions, which could lead to fake high results [72].

Table 5 shows the MSE results for all classifiers over nine data sets without preprocessing. For yeast-2_vs_4 dataset, our algorithm ranked first with the lowest MSE score (1.93E-04). For yeast-1_vs_7 dataset, our algorithm ranked second with score (0.031245), while the first algorithm AdaBoost achieved the lowest MSE score (0.028224). For yeast-1-4-5-8_vs_7 dataset, our algorithm ranked second with score (0.120345), while the first algorithm Naive Bayes achieved the lowest MSE score (0.114244). For yeast-2_vs_8 dataset, our algorithm ranked second with score (3.53E-02), while the first algorithm Naive Bayes achieved the lowest MSE score (0.032041). For yeast4 dataset, our algorithm ranked first with score (3.11E-05).

TABLE 5. MSE results without preprocessing.

data sets	Naive Bayes	AdaBoost	Decision Table	Random Tree	GWO+ MLP	WOA+ MLP	WOA+ BRNN	LSHGWO BRNN
yeast-2_vs_4	0.008464	6.3E-04	0.00504	0.014884	5.8E-04	3.8E-4	3.2E-04	**1.93E-04**
yeast-1_vs_7	0.039601	**0.028224**	0.336116	0.177241	0.040804	0.036524	0.035721	0.031245
yeast-1-4-5-8_vs_7	**0.114244**	0.132321	0.252004	0.249001	0.121428	0.126547	0.131769	0.120345
yeast-2_vs_8	**0.032041**	0.04	0.101124	0.149769	0.042849	3.60E-02	0.039601	3.53E-02
yeast4	0.021025	0.011449	0.194481	0.133956	0.010201	4.00E-04	0.008649	**3.11E-05**
yeast-1-2-8-9_vs_7	0.059536	0.076729	0.209764	0.137641	0.070756	**0.046235**	0.067081	0.049635
yeast5	2.0E-04	2.3E-04	0.031684	0.034596	2.6E-04	0.0002	**1.0E-04**	1.14E-04
yeast6	0.005929	0.007921	0.212521	0.078400	0.00922	3.00E-04	0.008100	**1.46E-04**
ECBDL 14	0.099856	0.082369	0.196249	0.173889	0.085849	0.084965	0.087025	**0.076958**
Average	0.042322	0.042208	0.170998	0.127709	0.042439	0.036839	0.042041	**0.034885**

For yeast-1-2-8-9_vs_7 dataset, our algorithm ranked second with score (0.049635), while the first algorithm WOA+MLP achieved the lowest MSE score (0.046235). For yeast5 dataset, our algorithm ranked second with score (1.14E-04), while the first algorithm WOA+BRNN achieved the lowest MSE score (1.0E-04). For yeast6 dataset, our algorithm ranked first with score (1.46E-04). For ECBDL 14 dataset, our algorithm ranked first with score (0.076958), while the first algorithm AdaBoost achieved (0.082369), then, GWO+MLP ranked second with score (0.085849). Finally, we achieved the lowest MSE score average among all other algorithms (0.034885), then, WOA+MLP algorithm ranked second with score (0.036839), then, WOA+BRNN algorithm ranked third with score (0.042041), then, AdaBoost algorithm ranked fourth with score (0.042208), then, Naive Bayes algorithm ranked fifth with score (0.042322),then, GWO+MLP algorithm ranked sixth with score (0.042439), then, Random Tree algorithm ranked seventh with score (0.127709), and finally, Decision Table algorithm ranked eighth with score (0.170998). Table 5 results are a clear evidence for high local optima avoidance ability of our proposed framework LSHGWOBRNN, even if the difference was not very big, but this actually is due to the fact, that all data sets in this stage were imbalanced and have not been preprocessed.

2) EXPERIMENT TWO (LSH-SMOTE PREPROCESSING)

In this experiment, we will compare the proposed algorithm LSHGWOBRNN against the seven classifiers (Naive Bayes, AdaBoostM1, Decision Table, Random Tree, GWO+MLP, WOA+MLP, and WOA+BRNN) over nine data sets preprocessed by LSH-SMOTE algorithm in terms of AUC and MSE (Local Optima Avoidance). The AUC results will be depicted in Table 6 while the MSE results will be depicted in Table 7.

Table 6 and Fig. 13 shows the AUC results for all classifiers over nine data sets preprocessed by LSH-SMOTE algorithm. For yeast-2_vs_4 dataset, our algorithm ranked first with score (0.996). For yeast-1_vs_7 dataset our algorithm ranked first with score (0.971). For yeast-1-4-5-8_vs_7 dataset our algorithm ranked first with score (0.979). For yeast-2_vs_8 dataset our algorithm ranked first with score (0.998). For yeast4 dataset, our algorithm ranked first with score (0.997). For yeast-1-2-8-9_vs_7 dataset our algorithm ranked first with score (0.995). For yeast5 dataset our algorithm ranked First with score (1). For yeast6 dataset our algorithm ranked first with score (0.999). For ECBDL 14 dataset, our algorithm ranked first with score (0.998). Finally, we achieved the best average over all other algorithms (0.993), then WOA+BRNN algorithm ranked second with score (0.987), then WOA+MLP algorithm ranked third with score (0.981), then GWO-MLP algorithm ranked fourth with score (0.980), then Decision Table algorithm ranked fifth with score (0.974), then Random Tree algorithm ranked sixth with score (0.945), then AdaBoost algorithm ranked seventh with score (0.908), and finally, Naive Bayes algorithm ranked eighth with score (0.879). The results of table 6 demonstrates that our proposed framework LSHGWOBRNN with preprocessing (LSH-SMOTE algorithm) has achieved best AUC score over all data sets, which is a clear evidence of the superior efficiency of the proposed algorithm over big imbalanced data sets. In contrary, the Naive Bayes algorithm failed and achieved the worst average among all other algorithms, due to its severe assumptions and poor quality of results.

Table 7 shows the MSE results for all classifiers over nine data sets preprocessed by LSH-SMOTE algorithm. For yeast-2_vs_4 dataset, our algorithm ranked first with the

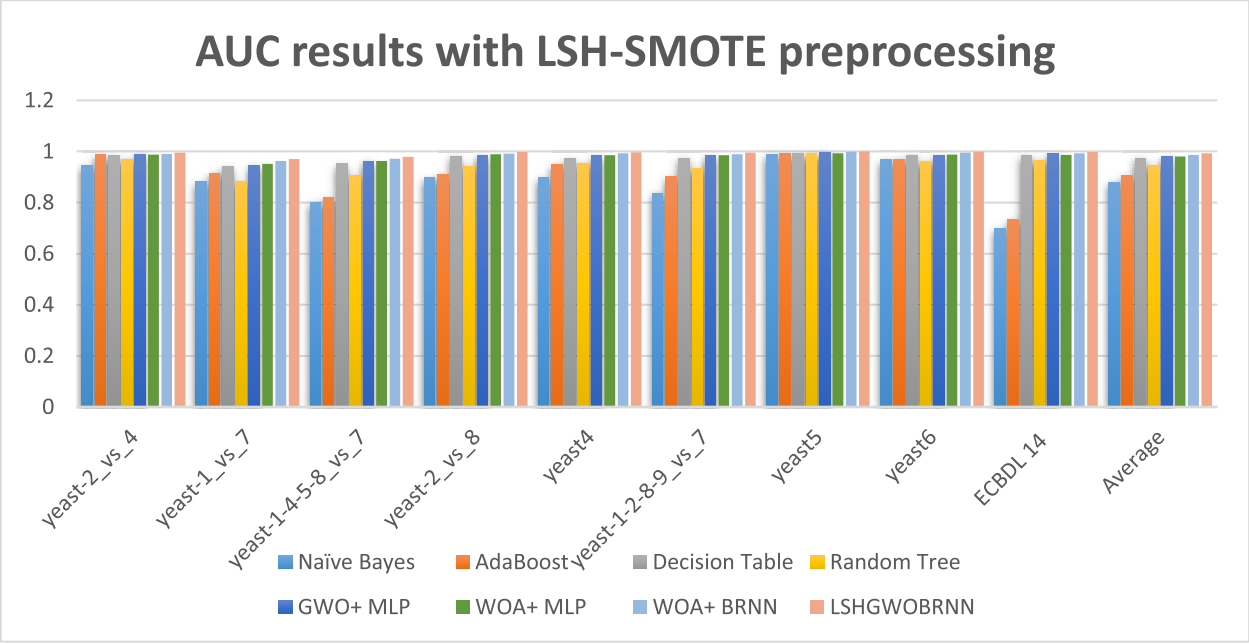

FIGURE 13. AUC results with LSH-SMOTE preprocessing.

TABLE 6. AUC results with lsh-smote preprocessing.

data sets	Naive Bayes	AdaBoost	Decision Table	Random Tree	GWO+ MLP	WOA+ MLP	WOA+ BRNN	LSHGWO BRNN
yeast-2_vs_4	0.944	0.988	0.983	0.967	0.989	0.988	0.991	**0.996**
yeast-1_vs_7	0.882	0.915	0.941	0.883	0.947	0.952	0.963	**0.971**
yeast-1-4-5-8_vs_7	0.801	0.819	0.952	0.906	0.962	0.963	0.972	**0.979**
yeast-2_vs_8	0.897	0.910	0.981	0.943	0.986	0.989	0.992	**0.998**
yeast4	0.898	0.949	0.972	0.954	0.983	0.986	0.993	**0.997**
yeast-1-2-8-9_vs_7	0.837	0.901	0.974	0.932	0.984	0.986	0.989	**0.995**
yeast5	0.989	0.991	0.991	0.992	0.996	0.993	0.999	**1**
yeast6	0.968	0.970	0.985	0.961	0.986	0.988	0.995	**0.999**
ECBDL 14	0.697	0.733	0.984	0.964	0.991	0.987	0.993	**0.998**
Average	0.879	0.908	0.974	0.945	0.980	0.981	0.987	**0.993**

lowest MSE score (1.39E-07). For yeast-1_vs_7 dataset our algorithm ranked first with the lowest MSE score (1.93E-06). For yeast-1-4-5-8_vs_7 dataset our algorithm ranked first with the lowest MSE score (3.11E-08). For yeast-2_vs_8 dataset our algorithm ranked first with the lowest MSE score (1.33E-07). For yeast4 dataset, our algorithm ranked first with the lowest MSE score (6.00E-09). For yeast-1-2-8-9_vs_7 dataset our algorithm ranked first with the lowest MSE score (3.83E-09). For yeast5 dataset our algorithm ranked First with the lowest MSE score (1.14E-09). For yeast6 dataset our algorithm ranked first with the lowest MSE score (1.46E-07). For ECBDL 14 dataset, our algorithm ranked first with the lowest MSE score (1.23E-06). Finally, we achieved the lowest MSE score average among all other algorithms (4.02E-07), then WOA+BRNN algorithm ranked second with score (0.000282), then WOA+MLP algorithm ranked third with score (0.0006),then GWO+MLP algorithm ranked fourth with score (0.000602), then Decision Table algorithm ranked fifth with score (0.000941), then Random Tree algorithm ranked sixth with score (0.004054), then

TABLE 7. MSE results with LSH-SMOTE preprocessing.

data sets	Naive Bayes	AdaBoost	Decision Table	Random Tree	GWO+ MLP	WOA+ MLP	WOA+ BRNN	LSHGWO BRNN
yeast-2_vs_4	0.003136	1.4E-04	2.9E-04	0.001089	1.2E-04	1.12E-4	8.1E-05	**1.39E-07**
yeast-1_vs_7	0.013924	0.007225	0.003481	0.013689	0.002809	0.002634	0.001369	**1.93E-06**
yeast-1-4-5-8_vs_7	0.039601	0.032761	0.002304	0.008836	0.001444	0.001357	7.8E-04	**3.11E-08**
yeast-2_vs_8	0.010609	0.0081	3.6E-04	0.003249	2.0E-04	1.71E-04	6.4E-05	**1.33E-07**
yeast4	0.010404	0.002601	7.8E-04	0.002116	2.9E-04	2.34E-04	4.9E-05	**6.00E-09**
yeast-1-2-8-9_vs_7	0.026569	0.009801	6.8E-04	0.004624	2.6E-04	2.27E-04	1.2E-04	**3.83E-09**
yeast5	1.2E-04	8.1E-05	8.1E-05	6.4E-05	1.6E-05	1.32E-05	1.0E-06	**1.14E-09**
yeast6	0.001024	9.0E-04	2.3E-04	0.001521	2.0E-04	1. 59E-04	2.5E-05	**1.46E-07**
ECBDL 14	0.091809	0.071289	2.6E-04	0.001296	8.1E-05	4.78E-05	4.9E-05	**1.23E-06**
Average	0.021911	0.014766	0.000941	0.004054	0.000602	0.0006	0.000282	**4.02E-07**

TABLE 8. Algorithm running time in minutes.

Data sets	GWO+ MLP	WOA+ MLP	WOA+ BRNN	LSHGWOBRNN
yeast-2_vs_4	0.637	0.621	0.800	**0.353**
yeast-1_vs_7	0.638	0.636	0.799	**0.414**
yeast-1-4-5-8_vs_7	1.175	1.157	1.526	**0.564**
yeast-2_vs_8	0.817	0.796	0.996	**0.377**
yeast4	2.238	2.183	2.997	**0.826**
yeast-1-2-8-9_vs_7	1.517	1.479	1.984	**0.661**
yeast5	2.337	2.272	3.064	**0.897**
yeast6	2.354	2.341	3.144	**0.974**
ECBDL 14	322.34	316.05	421	**139.97**

AdaBoost algorithm ranked seventh with score (0.014766), and finally, Naive Bayes algorithm ranked eighth with score (0.021911). From table 7 results, we can conclude that our proposed framework LSHGWOBRNN with preprocessing (LSH-SMOTE algorithm) has achieved the lowest MSE score over all data sets, which is a very strong evidence of the high local optima avoidance ability of our proposed framework LSHGWOBRNN over big imbalanced data sets.

3) COMPLEXITY AND oVERHEAD OF THE PROPOSED ALGORITHM ANALYSIS

Is this sub section, we will discuss the complexity and overhead of our proposed framework LSHGWOBRNN in terms of algorithm running time in minutes? We will measure the running time for (GWO+MLP, WOA+MLP, and WOA+BRNN) algorithms with each data set over all stages (preprocessing stage with LSH-SMOTE algorithm, and classification stage) and our proposed framework LSHGWOBRNN with each data set over all stages (preprocessing(LSH-SMOTE & hashing), bucket search, and final tournament winner). The results are illustrated in Table 8.

Table 8 depicts the running time of our proposed framework LSHGWOBRNN, as we can see from the table, the running time was reasonable for all data sets except for the ECBDL'14 big data set it was very long around 5 to 7 hours with all algorithms except for our algorithm, it was around 2 hours and 20 minutes. Hence, we can notice that our proposed

TABLE 9. Results of lshgwobrnn vs GWO+MLP and all other algorithms in (reference no.62) over four chosen data sets.

	Balloon dataset		Breast Cancer dataset		Iris dataset		Heart dataset	
Algorithm	**MSE**	**Accuracy**	**MSE**	**Accuracy**	**MSE**	**Accuracy**	**MSE**	**Accuracy**
LSHGWOBRNN	**5.87E-19**	**1.00**	**0.000763**	**1.00**	**0.00912**	**0.97**	**0.071671**	**0.91**
GWO+MLP	9.38E-15	1.00	0.0012	0.99	0.0229	0.91	0.1226	0.75
PSO+MLP	0.000585	1.00	0.034881	0.11	0.22868	0.37	0.188568	0.69
GA+MLP	5.08E-24	1.00	0.003026	0.98	0.089912	0.89	0.093047	0.59
ACO+MLP	0.004854	1.00	0.01351	0.40	0.405979	0.32	0.22843	0.00
ES+MLP	0.019055	1.00	0.04032	0.06	0.31434	0.46	0.192473	0.71
PBIL+MLP	2.49E-05	1.00	0.032009	0.07	0.116067	0.86	0.154096	0.45

framework LSHGWOBRNN has achieved a tangible saving in terms of running time as we have expected at the beginning. And this is the second contribution for our proposed framework LSHGWOBRNN after solving the local optima problem in high dimensional data sets.

C. SECOND EXPERIMENT

In this experiment, our proposed framework LSHGWOBRNN will be tested against the GWO+MLP and all other classifiers in that paper (Ant Colony Optimization (ACO), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Population-based Incremental Learning (PBIL), and Evolution Strategy (ES), over four data sets (Balloon, Breast cancer, Iris, and Heart) chosen from that paper [62] is terms of classification accuracy and MSE.

Table 9 illustrates the Results of LSHGWOBRNN vs GWO+MLP and all other algorithms in [62] over four chosen data sets. The First data set is the Balloon data set, which is considered as one of the easiest data sets, accordingly, all algorithms achieved full accuracy score easily. The second one is the Beast Cancer data set, which is known as difficult one, our proposed framework LSHGWOBRNN ranked first with the highest accuracy (1.00) and lowest MSE (0.000763), then, the GWO+MLP algorithm ranked second with accuracy (0.99) and MSE (0.0012). The third data set is the Iris dataset, which is more difficult than the Beast Cancer data set, our proposed framework LSHGWOBRNN ranked first with the highest accuracy (0.97) and lowest MSE (0.00912), then, the GWO+MLP ranked second with accuracy (0.91) and MSE (0.0229). The fourth and last data set is the Heart dataset, which is the most difficult among all data sets is this experiment, our proposed framework LSHGWOBRNN ranked first with the highest accuracy (0.91) and lowest MSE (0.071671), then, the GWO+MLP ranked second with accuracy (0.75) and MSE (0.1226). From previous results, we can conclude that, our proposed framework LSHGWOBRNN has achieved the best efficiency in terms of the high classification accuracy, and the best local optima avoidance with the lowest MSE results among all other classifiers. At the end, the excellent results of our proposed framework LSHGWOBRNN in all experiments are because of two main reasons; the first reason is, the hashing of the big data sets by using the LSH technique created smaller sub data sets in each bucket, accordingly, the GWO+BRNN was applied to less number of dimensions and instances in each bucket, which gave GWO+BRNN the ability of finding the top best 5 instances in each bucket more effectively and precisely. The second reason, is our tournament technique (best of the best), in other words, after finding the top 5 instances in each bucket, we escalated them to the final stage of our tournament creating a new data set that contains all buckets winners called the final data set, after that the GWO+BRNN technique was applied to this final data set for searching for the best among them all (the tournament winner).

V. CONCLUSION

In this paper, a novel framework for imbalanced big data sets LSHGWOBRNN is presented. The proposed framework consists of three stages. The first stage uses the LSH-SMOTE algorithm for solving the class imbalance problem, then, it uses the LSH algorithm for hashing the data set into buckets. In the second stage, we used the GWO optimizer for finding the optimal values for weights and biases of the BRNN classifier, then, use our proposed algorithm GWO+BRNN for finding the best 5 search agents in each bucket in terms of MSE scores, then, composing a new data set from all bucket's winners named Finaldataset. In the third and last stage, GWO+BRNN algorithm is used for finding the tournament winner (the best instance among all the bests from all buckets in terms of the lowest MSE score). Our experimental results proved that our proposed framework can effectively and efficiently handle extremely imbalanced big datasets and achieving better classification results for all datasets. For testing the validity of our proposed LSHGWOBRNN framework, we performed two main experiments. In the first

experiment, we tested our proposed framework LSHGWOBRNN against seven classifiers over nine imbalanced datasets in terms of AUC and MSE with and without preprocessing and we achieved the best AUC average over all algorithms (0.848) and the lowest MSE among all algorithms (0.034885) without preprocessing, and, when the datasets were preprocessed by LSH-SMOTE algorithm, we achieved the best AUC average over all algorithms (0.993) and the Best MSE over all algorithms (4.02E-07) respectively. Additionally, our proposed framework LSHGWOBRNN has achieved tangible saving in terms of running time against other algorithms. In the second experiment, we tested our proposed framework LSHGWOBRNN against the GWO+MLP and all other classifiers in that papers (PSO, GA, ACO, ES, and PBIL), over four data sets (Balloon, Breast cancer, Iris, and Heart) chosen from that paper, and we have achieved the highest accuracy and lowest MSE at the same time over all data sets, which is a clear evidence for the high local optima avoidance ability of our proposed framework. At the end, we can conclude that our proposed framework LSHGWOBRNN can handle extremely imbalanced big data sets and achieve better classification results.

FUTURE WORK

In our proposed algorithm LSHGWOBRNN we achieved two important improvements (local optima elimination and huge saving in running time), but there was also a limitation, which is the long running time for the big data set ECBDL'14, although that we have performed all experiments on the test set only not the whole dataset. At the beginning, we tried to load the whole dataset into Weka 3.9.1 for applying the LSH-SMOTE, but the Weka stopped working. After that, we tried to load the test set only without reducing the number of attributes, and again the Weka stopped working. Accordingly, we have been forced to reduce the number of attributes to get it work, and after all of that, the running time was a little bit long, it took more than two hours to finish. To overcome such issues in our future work, we are looking for a distributed version of our algorithm LSHGWOBRNN based on Spark or Hadoop for processing such big datasets without any issues and with a reasonable running time.

REFERENCES

[1] E. Ahmed, I. Yaqoob, I. A. T. Hashem, J. Shuja, M. Imran, N. Guizani, and S. T. Bakhsh, "Recent advances and challenges in mobile big data," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 102–108, Feb. 2018.

[2] Y. Wang, L. Kung, and T. A. Byrd, "Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations," *Technol. Forecasting Social Change*, vol. 126, pp. 3–13, Jan. 2018.

[3] K. F. Tiampo, S. McGinnis, Y. Kropivnitskaya, J. Qin, and M. A. Bauer, "Big data challenges and hazards modeling," in *Risk Modeling for Hazards and Disasters*. Amsterdam, The Netherlands: Elsevier, 2018, pp. 193–210.

[4] I. El Alaoui, Y. Gahi, and R. Messoussi, "Full consideration of big data characteristics in sentiment analysis context," in *Proc. IEEE 4th Int. Conf. Cloud Comput. Big Data Anal. (ICCCBDA)*, Apr. 2019, pp. 126–130.

[5] C. E. Turcu and C. O. Turcu, "New perspectives on sustainable healthcare delivery through Web of things," in *Handbook of Research on Contemporary Perspectives on Web-Based Systems*. Hershey, PA, USA: IGI Global, 2018, pp. 166–187.

[6] P. Chaudhary and B. B. Gupta, "A novel framework to alleviate dissemination of XSS worms in online social network (OSN) using view segregation," *Neural Netw. World*, vol. 27, no. 1, p. 5, 2017.

[7] V. Grover, R. H. L. Chiang, T.-P. Liang, and D. Zhang, "Creating strategic business value from big data analytics: A research framework," *J. Manage. Inf. Syst.*, vol. 35, no. 2, pp. 388–423, 2018.

[8] B. N. Silva, M. Khan, and K. Han, "Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities," *Sustain. Soc.*, vol. 38, pp. 697–713, Apr. 2018.

[9] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constant, and K. Mankodiya, "Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare," *Future Generat. Comput. Syst.*, vol. 78, pp. 659–676, Jan. 2018.

[10] E. Ascarza, S. A. Neslin, O. Netzer, Z. Anderson, P. S. Fader, S. Gupta, B. G. S. Hardie, A. Lemmens, B. Libai, D. Neal, F. Provost, and R. Schrift, "In pursuit of enhanced customer retention management: Review, key issues, and future directions," *Customer Needs Solutions*, vol. 5, nos. 1–2, pp. 65–81, 2018.

[11] S. S. Reka and T. Dragicevic, "Future effectual role of energy delivery: A comprehensive review of Internet of Things and smart grid," *Renew. Sustain. Energy Rev.*, vol. 91, pp. 90–108, Aug. 2018.

[12] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 97–107, Jan. 2014.

[13] F. Amalina, I. A. T. Hashem, Z. H. Azizul, A. T. Fong, A. Firdaus, M. Imran, and N. B. Anuar, "Blending big data analytics: Review on challenges and a recent study," *IEEE Access*, to be published.

[14] H. Kaur, M. A. Alam, R. Jameel, A. K. Mourya, and V. Chang, "A proposed solution and future direction for blockchain-based heterogeneous medicare data in cloud environment," *J. Med. Syst.*, vol. 42, no. 8, p. 156, Aug. 2018.

[15] W. Wu, W. Lin, C.-H. Hsu, and L. He, "Energy-efficient hadoop for big data analytics and computing: A systematic review and research insights," *Future Gener. Comput. Syst.*, vol. 86, pp. 1351–1367, Sep. 2018.

[16] S. García, Z.-L. Zhang, A. Altalhi, S. Alshomrani, and F. Herrera, "Dynamic ensemble selection for multi-class imbalanced datasets," *Inf. Sci.*, vols. 445–446, pp. 22–37, Jun. 2018.

[17] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Evanston, IL, USA: Routledge, 2018.

[18] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Netw.*, vol. 106, pp. 249–259, Oct. 2017.

[19] S. Ramírez-Gallego, A. Fernández, S. García, M. Chen, and F. Herrera, "Big data: Tutorial and guidelines on information and process fusion for analytics algorithms with MapReduce," *Inf. Fusion*, vol. 42, pp. 51–61, Jul. 2018.

[20] P. J. Werbos, "The new AI: Basic concepts, and urgent risks and opportunities in the Internet Of Things," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. New York, NY, USA: Academic, 2019, pp. 161–190.

[21] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Inf. Fusion*, vol. 42, pp. 146–157, Jul. 2018.

[22] Accessed: Apr. 2019. [Online]. Available: https://www.nvidia.com/en-us/geforce/news/graphics-reinvented-new-technologies-in-rtx-graphics-cards/

[23] M. Amer and T. Maul, "A review of modularization techniques in artificial neural networks," *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 527–561, 2019.

[24] P. O'Connor, E. Gavves, and M. Welling, "Training a spiking neural network with equilibrium propagation," in *Proc. 22nd Int. Conf. Artif. Intell. Statist.*, 2019, pp. 1516–1523.

[25] T. Chakraborty, S. Chattopadhyay, and A. K. Chakraborty, "A novel hybridization of classification trees and artificial neural networks for selection of students in a business school," *OPSEARCH*, vol. 55, no. 2, pp. 434–446, 2018.

[26] P. Singh and P. Dwivedi, "Integration of new evolutionary approach with artificial neural network for solving short term load forecast problem," *Appl. Energy*, vol. 217, pp. 537–549, May 2018.

[27] I. Aljarah, H. Faris, and S. Mirjalili, "Optimizing connection weights in neural networks using the whale optimization algorithm," *Soft Comput.*, vol. 22, no. 1, pp. 1–15, 2016.

[28] I. Aljarah, H. Faris, S. Mirjalili, and N. Al-Madi, "Training radial basis function networks using biogeography-based optimizer," *Neural Comput. Appl.*, vol. 29, no. 7, pp. 529–553, 2018.

[29] P. Yin, M. Pham, A. Oberman, and S. Osher, "Stochastic backward Euler: An implicit gradient descent algorithm for k-means clustering," *J. Sci. Comput.*, vol. 77, no. 2, pp. 1133–1146, 2018.

[30] A. Saad, S. A. Khan, and A. Mahmood, "A multi-objective evolutionary artificial bee colony algorithm for optimizing network topology design," *Swarm Evol. Comput.*, vol. 38, pp. 187–201, 2018.
[31] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.
[32] S. Piri, D. Delen, and T. Liu, "A synthetic informative minority over-sampling (SIMO) algorithm leveraging support vector machine to enhance learning from imbalanced datasets," *Decis. Support Syst.*, vol. 106, pp. 15–29, Feb. 2018.
[33] M. Pagano and K. Gauvreau, *Principles of Biostatistics*. Boca Raton, FL, USA: CRC Press, 2018.
[34] S. Das, S. Datta, and B. B. Chaudhuri, "Handling data irregularities in classification: Foundations, trends, and future challenges," *Pattern Recognit.*, vol. 81, pp. 674–693, Sep. 2018.
[35] A. Alonso-Betanzos, V. Bolón-Canedo, C. Eiras-Franco, L. Morán-Fernández, and B. Seijo-Pardo, "Preprocessing in high dimensional datasets," in *Advances in Biomedical Informatics*. Cham, Switzerland: Springer, 2018, pp. 247–271.
[36] F. Hu, C. Yu, J. Dai, and K. Liu, "A mixed sampling method for imbalanced data based on neighborhood density," in *Proc. IEEE 4th Int. Conf. Cloud Comput. Big Data Anal. (ICCCBDA)*, Apr. 2019, pp. 94–98.
[37] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.
[38] M. Aydar and S. Ayvaz, "An improved method of locality-sensitive hashing for scalable instance matching," *Knowl. Inf. Syst.*, vol. 58, no. 2, pp. 275–294, 2019.
[39] S.-A. N. Alexandropoulos, C. K. Aridas, S. B. Kotsiantis, and M. N. Vrahatis, "Multi-objective evolutionary optimization algorithms for machine learning: A recent survey," in *Approximation and Optimization*. Cham, Switzerland: Springer, 2019, pp. 35–55.
[40] C. Yu, L. Luo, L. L.-H. Chan, T. Rakthanmanon, and S. Nutanong, "A fast LSH-based similarity search method for multivariate time series," *Inf. Sci.*, vol. 476, pp. 337–356, Feb. 2019.
[41] H. A. Dau, D. F. Silva, F. Petitjean, G. Forestier, A. Bagnall, A. Mueen, and E. Keogh, "Optimizing dynamic time warping's window width for time series data mining applications," *Data Mining Knowl. Discovery*, vol. 32, no. 4, pp. 1074–1120, 2018.
[42] A. Darwish, A. E. Hassanien, and S. Das, "A survey of swarm and evolutionary computing approaches for deep learning," *Artif. Intell. Rev.*, pp. 1–46, Jun. 2019.
[43] Z. Chen, X. He, J. Sun, H. Chen, and L. He, "Concurrent hash tables on multicore machines: Comparison, evaluation and implications," *Future Gener. Comput. Syst.*, vol. 82, pp. 127–141, May 2018.
[44] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 13th Annu. ACM Symp. Theory Comput.*, 1998, pp. 604–613.
[45] M. Bury, C. Schwiegelshohn, and M. Sorella, "Similarity search for dynamic data streams," *IEEE Trans. Knowl. Data Eng.*, to be published.
[46] Z. M. M. Aye, B. I. P. Rubinstein, and K. Ramamohanarao, "Fast manifold landmarking using locality-sensitive hashing," in *Proc. Pacific–Asia Conf. Knowl. Discovery Data Mining*. Cham, Switzerland: Springer, 2018, pp. 452–464.
[47] L. Paulevé, H. Jégou, and L. Amsaleg, "Locality sensitive hashing: A comparison of hash function types and querying mechanisms," *Pattern Recognit. Lett.*, vol. 31, no. 11, pp. 1348–1358, Aug. 2010.
[48] W. Hu, Y. Fan, J. Xing, L. Sun, Z. Cai, and S. Maybank, "Deep constrained siamese hash coding network and load-balanced locality-sensitive hashing for near duplicate image detection," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4452–4464, Sep. 2018.
[49] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997.
[50] D. L. Minh, A. Sadeghi-Niaraki, H. D. Huy, K. Min, and H. Moon, "Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network," *IEEE Access*, vol. 6, pp. 55392–55404, 2018.
[51] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Netw.*, vol. 107, pp. 3–11, Nov. 2018.
[52] M. Berglund, T. Raiko, M. Honkala, L. Kärkkäinen, A. Vetek, and J. T. Karhunen, "Bidirectional recurrent neural networks as generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 856–864.
[53] M. Duggan, R. Shaw, J. Duggan, E. Howley, and E. Barrett, "A multitime-steps-ahead prediction approach for scheduling live migration in cloud data centers," *Softw., Pract. Exper.*, vol. 49, no. 4, pp. 617–639, 2019.
[54] T. F. Kurnaz and Y. Kaya, "The comparison of the performance of ELM, BRNN, and SVM methods for the prediction of compression index of clays," *Arabian J. Geosci.*, vol. 11, no. 24, p. 770, 2018.
[55] L. K. Sharma, V. Vishal, and T. N. Singh, "Developing novel models using neural networks and fuzzy systems for the prediction of strength of rocks from key geomechanical properties," *Measurement*, vol. 102, pp. 158–169, May 2017.
[56] A. L. Caterini and D. E. Chang, *Deep Neural Networks in a Mathematical Framework*. Oxford, U.K.: Springer, 2018.
[57] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[58] C. Muro, R. Escobedo, L. Spector, and R. P. Coppinger, "Wolf-pack *(Canis lupus)* hunting strategies emerge from simple rules in computational simulations," *Behav. Processes*, vol. 88, no. 3, pp. 192–197, 2011.
[59] E. M. Hassib, A. E. Eldesokey, L. M. Labib, and S. Elghamrawy, "LSH-SMOTE: A modified SMOTE algorithm for imbalanced data-sets," *Ciência Técnica Vitivinícola*, vol. 33, no. 4, pp. 50–65, 2018.
[60] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 372–387.
[61] A. Ragni, Q. Li, M. J. F. Gales, and Y. Wang, "Confidence estimation and deletion prediction using bidirectional recurrent neural networks," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Dec. 2018, pp. 204–211.
[62] S. Mirjalili, "How effective is the Grey Wolf optimizer in training multi-layer perceptrons," *Appl. Intell.*, vol. 43, no. 1, pp. 150–161, 2015.
[63] *KEEL-Dataset Repository*. Accessed: Jun. 2019. [Online]. Available: http://www.keel.es/dataset.php
[64] *Data Mining Competition 2014: Self-Deployment Track*. Accessed: May 2019. [Online]. Available: http://cruncher.ico2s.org/bdcomp/
[65] D. Martens, J. Vanthienen, W. Verbeke, and B. Baesens, "Performance of classification models from a user perspective," *Decis. Support Syst.*, vol. 51, no. 4, pp. 782–793, 2011.
[66] N. V. Chawla, "Data mining for imbalanced datasets: An overview," in *Data Mining and Knowledge Discovery Handbook*. Boston, MA, USA: Springer, 2009, pp. 875–886.
[67] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Inf. Sci.*, vol. 250, pp. 113–141, Nov. 2013.
[68] W. Lee, C.-H. Jun, and J.-S. Lee, "Instance categorization by support vector machines to adjust weights in AdaBoost for imbalanced data classification," *Inf. Sci.*, vol. 381, pp. 92–103, Mar. 2017.
[69] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification," *Pattern Recognit. Lett.*, vol. 30, no. 1, pp. 27–38, 2009.
[70] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 42, no. 4, pp. 1119–1130, Aug. 2012.
[71] E. M. Hassib, A. I. El-Desouky, L. M. Labib, and E.-S. M. El-kenawy, "WOA + BRNN: An imbalanced big data classification framework using Whale optimization and deep neural network," *Soft Comput.*, pp. 1–20, Mar. 2019.
[72] J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger, "Tackling the poor assumptions of naive Bayes text classifiers," in *Proc. 20th Int. Conf. Mach. Learn. (ICML)*, 2003, pp. 616–623.

ESLAM MOHSEN HASSIB is currently pursuing the Ph.D. degree in computer engineering with the Faculty of Engineering Department, Mansoura University, Egypt. He is currently working on a new framework for imbalanced big data mining. His research areas include big data analysis, imbalanced big data classification, spark, Hadoop, optimization algorithms, deep learning, and big data frameworks.

ALI IBRAHIM EL-DESOUKY received the M.A. and Ph.D. degrees from the University of Glasgow, USA. He is currently a Full Professor with the Computers Engineering and Systems Department at the Faculty of Engineering, Mansoura University, Egypt. He is also a visiting part-time Professor with MET Academy. He teaches in American and Mansoura universities, and has taken over many positions of leadership and supervision of many scientific papers. He has published hundreds of articles in well-known international journals.

EL-SAYED M. EL-KENAWY is currently an Assistant Professor with the Delta Higher Institute for Engineering & Technology (DHIET), Mansoura, Egypt, where he was inspiring and motivating students by providing a thorough understanding of a variety of computer concepts. He has pioneered and launched independent research programs. He is interested in computer science and machine learning field. He is adept at explaining sometimes complex concepts in an easy-to-understand manner.

SALLY M. EL-GHAMRAWY received the B.Sc. degree in computers engineering and systems, in 2003, and the M.Sc. degree in automatic control systems engineering and the Ph.D. degree in distributed decision support systems based on multi intelligent agents from the Computer Engineering Department, Faculty of Engineering, Mansoura University, Egypt, in 2006 and 2012, respectively. She is currently the Head of the Communications and Computer Engineering Department, MISR Higher Institute for Engineering and Technology, and a part-time Associate Professor with the Electrical and Computers Engineering Department, Faculty of Engineering, The British University in Egypt (BUE), and at the Computers Engineering Department, Faculty of Engineering, Mansoura University. She was delivering lectures, as well as supervising graduation projects, master's thesis, and doctoral dissertations. She was delivering lectures and gave practical training in the grants from the Ministry of Communications and Information Technology, with the collaboration of IBM. She received a certificate A+ International Inc., CompTIA. She is a member of the Scientific Research Group, Egypt. Her research focuses on big data analysis, No-SQL databases, Hadoop, MapReduce techniques, and software engineering. She is the author of number peer-reviewed publications, receiving the best paper awards. She is a reviewer of a number of international journals, and a judge on the IEEE Young Professionals' Competitions.